

chapter 7

Public Key Infrastructure

Without trust, there is nothing.

—ANONYMOUS



In this chapter, you will learn how to

- Implement the basics of public key infrastructures
- Describe the roles of certificate authorities and certificate repositories
- Explain the relationship between trust and certificate verification
- Identify centralized and decentralized infrastructures
- Understand the lifecycle of certificates
- Describe public and in-house certificate authorities
- Identify the standards involved in establishing an interoperable Internet PKI
- Explain interoperability issues with PKI standards
- Describe how the common Internet protocols implement the PKI standards

Public key infrastructures (PKIs) are becoming a central security foundation for managing identity credentials in many companies. The technology manages the issue of binding public keys and identities across multiple applications. The other approach, without PKIs, is to implement many different security solutions and hope for interoperability and equal levels of protection.

PKIs comprise several components, including certificates, registration and certificate authorities, and a standard process for verification. PKIs are about managing the sharing of trust and using a third party to vouch for the trustworthiness of a claim of ownership over a credential document, called a **certificate**.

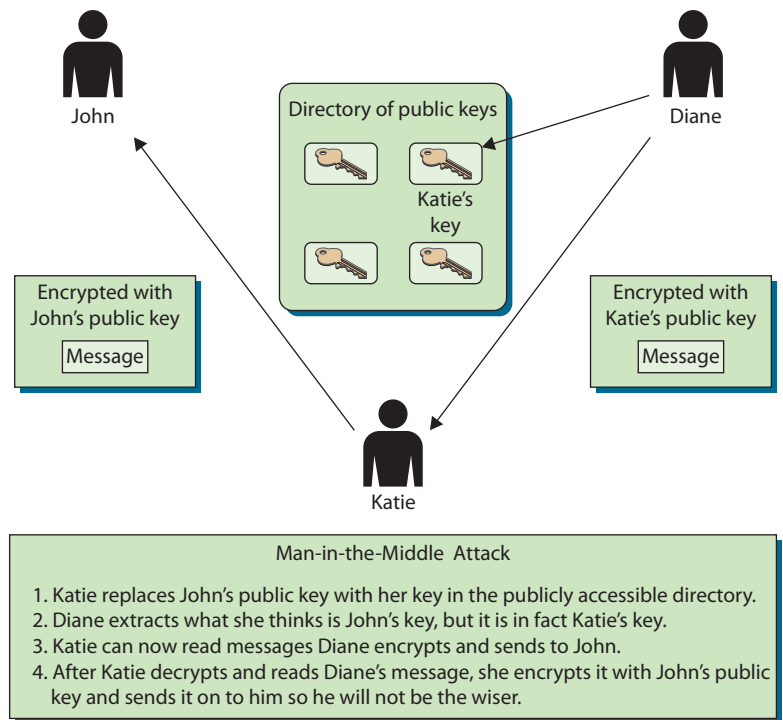
■ The Basics of Public Key Infrastructures

A **public key infrastructure (PKI)** provides all the components necessary for different types of users and entities to be able to communicate securely and in a predictable manner. A PKI is made up of hardware, applications, policies, services, programming interfaces, cryptographic algorithms, protocols, users, and utilities. These components work together to allow communication to take place using public key cryptography and symmetric keys for digital signatures, data encryption, and integrity.

Although many different applications and protocols can provide the same type of functionality, constructing and implementing a PKI boils down to establishing a level of trust. If, for example, John and Diane want to communicate securely, John can generate his own public/private key pair and send his public key to Diane, or he can place his public key in a directory that is available to everyone. If Diane receives John's public key, either from him or from a public directory, how does she know the key really came from John? Maybe another individual, Katie, is masquerading as John and has replaced John's public key with her own, as shown in Figure 7.1 (referred to as a man-in-the-middle attack). If this took place, Diane would believe that her messages could be read only by John and that the replies were actually from him. However, she would actually be communicating with Katie. What is needed is a way to verify an individual's identity, to ensure that a person's public key is bound to their identity and thus ensure that the previous scenario (and others) cannot take place.

In PKI environments, entities called registration authorities (RAs) and certificate authorities (CAs) provide services similar to those of the Department of Motor Vehicles (DMV). When John goes to register for a driver's license, he has to prove his identity to the DMV by providing his passport, birth certificate, or other identification documentation. If the DMV is satisfied with the proof John provides (and John passes a driving test), the DMV will create a driver's license that can then be used by John to prove his identity. Whenever John needs to identify himself, he can show his driver's license. Although many people may not trust John to identify himself truthfully, they do trust the third party, the DMV.

In the PKI context, while some variations exist in specific products, the RA will require proof of identity from the individual requesting a certificate and will validate this information. The RA will then advise the CA to generate a certificate, which is analogous to a driver's license. The CA will digitally sign the certificate using its private key. The use of the private key ensures to the recipient that the certificate came from the CA. When Diane receives John's certificate and verifies that it was actually digitally signed



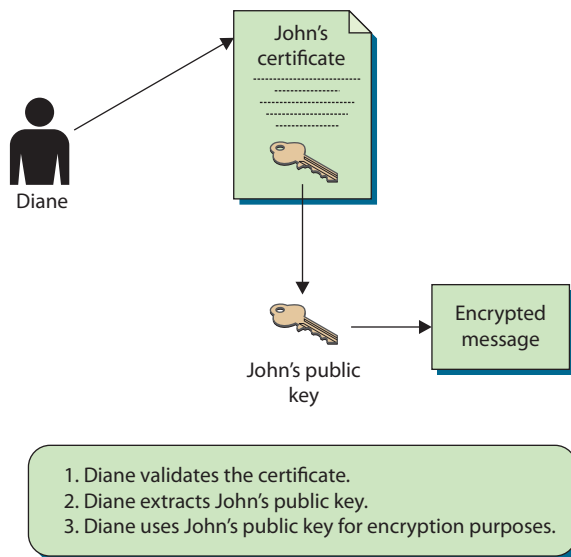
• **Figure 7.1** Without PKIs, individuals could spoof others' identities.



Tech Tip

Public and Private Keys

Recall from Chapter 5 that the public key is the one you give to others and that the private key never leaves your possession. Anything one key does, the other undoes, so if you encrypt something with the public key, only the holder of the private key can decrypt it. If you encrypt something with the private key, then everyone who uses the public key knows that the holder of the private key did the encryption. Certificates do not alter any of this; they only offer a standard means of transferring keys.



• **Figure 7.2** Public keys are components of digital certificates.



PKIs are composed of several elements:

- Certificates (containing keys)
- Certificate authorities (CAs)
- Registration authorities (RAs)
- Certificate revocation lists (CRLs)
- Trust models

by a CA that she trusts, she will believe that the certificate is actually John's—not because she trusts John, but because she trusts the entity that is vouching for his identity (the CA).

This is commonly referred to as a *third-party trust model*. Public keys are components of digital certificates, so when Diane verifies the CA's digital signature, this verifies that the certificate is truly John's and that the public key the certificate contains is also John's. This is how John's identity is bound to his public key.

This process allows John to authenticate himself to Diane and others. Using the third-party certificate, John can communicate with Diane, using public key encryption, without prior communication or a preexisting relationship.

Once Diane is convinced of the legitimacy of John's public key, she can use it to encrypt messages between herself and John, as illustrated in Figure 7.2.

Numerous applications and protocols can generate public/private key pairs and provide functionality similar to what a PKI provides, but no trusted third party is available for both of the communicating parties. For each party to choose to communicate this way without a third party vouching for the other's identity, the two must choose to trust each other and the communication channel they are using. In many situations, it is impractical and dangerous to arbitrarily trust an individual you do not know, and this is when the components of a PKI must fall into place—to provide the necessary level of trust you cannot, or choose not to, provide on your own.

What does the “infrastructure” in public key infrastructure really mean? An infrastructure provides a sustaining groundwork upon which other things can be built. So an infrastructure works at a low level to provide a predictable and uniform environment that allows other, higher-level technologies to work together through uniform access points. The environment that the infrastructure provides allows these higher-level applications to communicate with each other and gives them the underlying tools to carry out their tasks.

■ Certificate Authorities

A **certificate authority (CA)** is a trusted authority that certifies individuals' identities and creates electronic documents indicating that individuals are who they say they are. The electronic document is referred to as a **digital certificate**, and it establishes an association between the subject's identity and a public key. The private key that is paired with the public key in the certificate is stored separately.

A CA is more than just a piece of software, however; it is actually made up of the software, hardware, procedures, policies, and people who are involved in validating individuals' identities and generating the certificates. This means that if one of these components is compromised, it can

negatively affect the CA overall and can threaten the integrity of the certificates it produces.



Cross Check

Certificates Stored on a Client PC

Certificates are stored on user PCs. Chapter 17 covers the use of the Internet and associated materials, including the use of certificates by web browsers. Take a moment to explore the certificates stored on your PC by your browser. To understand the details behind how certificates are stored and managed, see the details in Chapter 17.

Every CA should have a **certification practices statement (CPS)** that outlines how identities are verified; the steps the CA follows to generate, maintain, and transmit certificates; and why the CA can be trusted to fulfill its responsibilities.

The CPS describes how keys are secured, what data is placed within a digital certificate, and how revocations will be handled. If a company is going to use and depend on a public CA, the company's security officers, administrators, and legal department should review the CA's entire CPS to ensure that it will properly meet the company's needs, and to make sure that the level of security claimed by the CA is high enough for their use and environment. A critical aspect of a PKI is the trust between the users and the CA, so the CPS should be reviewed and understood to ensure that this level of trust is warranted.

The **certificate server** is the actual service that issues certificates based on the data provided during the initial registration process. The server constructs and populates the digital certificate with the necessary information and combines the user's public key with the resulting certificate. The certificate is then digitally signed with the CA's private key.



Tech Tip

Trusting CAs

The question of whether a CA can be trusted is part of the continuing debate on how much security PKIs actually provide. Overall, people put a lot of faith in CAs. If a CA was compromised or did not follow through on its various responsibilities, word would get out and it would quickly lose customers and business. CAs work diligently to ensure that the reputation of their products and services remains good by implementing very secure facilities, methods, procedures, and trained personnel. But it is up to the company or individual to determine what degree of trust can actually be given and what level of risk is acceptable.

Registration Authorities

A **registration authority (RA)** is the PKI component that accepts a request for a digital certificate and performs the necessary steps of registering and authenticating the person requesting the certificate. The authentication requirements differ depending on the type of certificate being requested. Most CAs offer a series of classes of certificates with increasing trust by class. The specific classes are described later in the section titled "Certificate Classes."

Each higher class of certificate can carry out more powerful and critical tasks than the one below it. This is why the different classes have different requirements for proof of identity. If you want to receive a Class 1 certificate, you may only be asked to provide your name, e-mail address, and physical address. For a Class 2 certification, you may need to provide the RA with more data, such as your driver's license, passport, and company information that can be verified. To obtain a Class 3 certificate, you will be asked to provide even more information and most likely will need to go to the RA's office for a face-to-face meeting. Each CA will outline the certification

classes it provides and the identification requirements that must be met to acquire each type of certificate.

Local Registration Authorities

A **local registration authority (LRA)** performs the same functions as an RA, but the LRA is closer to the end users. This component is usually implemented in companies that have their own internal PKIs and have distributed sites. Each site has users that need RA services, so instead of requiring them to communicate with one central RA, each site can have its own LRA. This reduces the amount of traffic that would be created by several users making requests across wide area network (WAN) lines. The LRA performs identification, verification, and registration functions. It then sends the request, along with the user's public key, to a centralized CA so that the certificate can be generated. It acts as an interface between the users and the CA. LRAs simplify the RA/CA process for entities that desire certificates only for in-house use.

Public Certificate Authorities

An individual or company might decide to rely on a CA that is already established and being used by many other individuals and companies—a public CA. Alternatively, the company might decide it needs its own CA for internal use, which gives the company more control over the certificate registration and generation process and allows it to configure items specifically for its own needs. This second type of CA is referred to as a *private CA* (or *in-house CA*), discussed in the next section.

A public CA specializes in verifying individual identities and creating and maintaining their certificates. These companies issue certificates that are not bound to specific companies or intracompany departments. Instead, their services are to be used by a larger and more diversified group of people and organizations. If a company uses a public CA, the company will pay the CA organization for individual certificates and for the service of maintaining these certificates. Some examples of public CAs are VeriSign (including GeoTrust and Thawte), Entrust, and GoDaddy.

One advantage of using a public CA is that it is usually well known and easily accessible to many people. Most web browsers have a list of public CAs installed and configured by default, along with their corresponding root certificates. This means that if you install a web browser on your computer, it is already configured to trust certain CAs, even though you might have never heard of them before. So, if you receive a certificate from Bob, and his certificate was digitally signed by a CA listed in your browser, you automatically trust the CA and can easily walk through the process of verifying Bob's certificate. This has raised some eyebrows among security professionals, however, since trust is installed by default, but the industry has deemed this is a necessary approach that provides users with transparency and increased functionality.

The *certificate policy (CP)* allows users to decide what certification classes are acceptable and how they will be used within the organization. This is different from the CPS, which explains how the CA verifies entities, generates certificates, and maintains these certificates. The CP is generated and



Users can remove CAs from their browser list if they want to have more control over whom their system trusts and doesn't trust. Unfortunately, system updates can restore the CAs, thus requiring regular certificate store maintenance on the part of the users.

owned by an individual company that uses an external CA, and it allows the company to enforce *its* security decisions and control how certificates are used with its applications.

In-house Certificate Authorities

An *in-house* CA is implemented, maintained, and controlled by the company that implemented it. This type of CA can be used to create certificates for internal employees, devices, applications, partners, and customers. This approach gives the company complete control over how individuals are identified, what certification classifications are created, who can and cannot have access to the CA, and how the certifications can be used.

Choosing Between a Public CA and an In-house CA

When deciding between an in-house CA and public CA, you need to identify and account for various factors. Setting up your own PKI takes significant resources, especially skilled personnel. Several companies have started on a PKI implementation, only to quit halfway through, resulting in wasted time and money, with nothing to show for it except heaps of frustration and many ulcers.

In some situations, it is better for a company to use a public CA, since public CAs already have the necessary equipment, skills, and technologies. In other situations, companies may decide it is a better business decision to take on these efforts themselves. This is not always a strictly monetary decision—a specific level of security might be required. Some companies do not believe they can trust an outside authority to generate and maintain their users' and company's certificates. In this situation, the scale may tip toward an in-house CA.

Each company is unique, with various goals, security requirements, functionality needs, budgetary restraints, and ideologies. The decision of whether to use a private CA or an in-house CA depends on the expansiveness of the PKI within the organization, how integrated it will be with different business needs and goals, its interoperability with the company's current technologies, the number of individuals who will be participating, and how it will work with outside entities. This could be quite a large undertaking that ties up staff, resources, and funds, so a lot of strategic planning is required, and what will and won't be gained from a PKI should be fully understood before the first dollar is spent on the implementation.

Outsourced Certificate Authorities

The last available option for using PKI components within a company is to outsource different parts of it to a specific service provider. Usually, the more complex parts are outsourced, such as the CA, RA, CRL, and key recovery mechanisms. This occurs if a company does not have the necessary skills to implement and carry out a full PKI environment.

Although outsourced services might be easier for your company to implement, you need to review several factors before making this type of



Tech Tip

Why In-house CAs?

In-house CAs provide more flexibility for companies, which often integrate them into current infrastructures and into applications for authentication, encryption, and nonrepudiation purposes. If the CA is going to be used over an extended period of time, this can be a cheaper method of generating and using certificates than having to purchase them through a public CA. Setting up in-house certificate servers is relatively easy and can be done with simple software that targets both Windows and Linux servers.



Certificate authorities come in many types: public, in-house, and outsourced. All of them perform the same functions, with the only difference being an organizational one. This can have a bearing on trust relationships, as one is more likely to trust in-house CAs over others for which there is arguably less control.



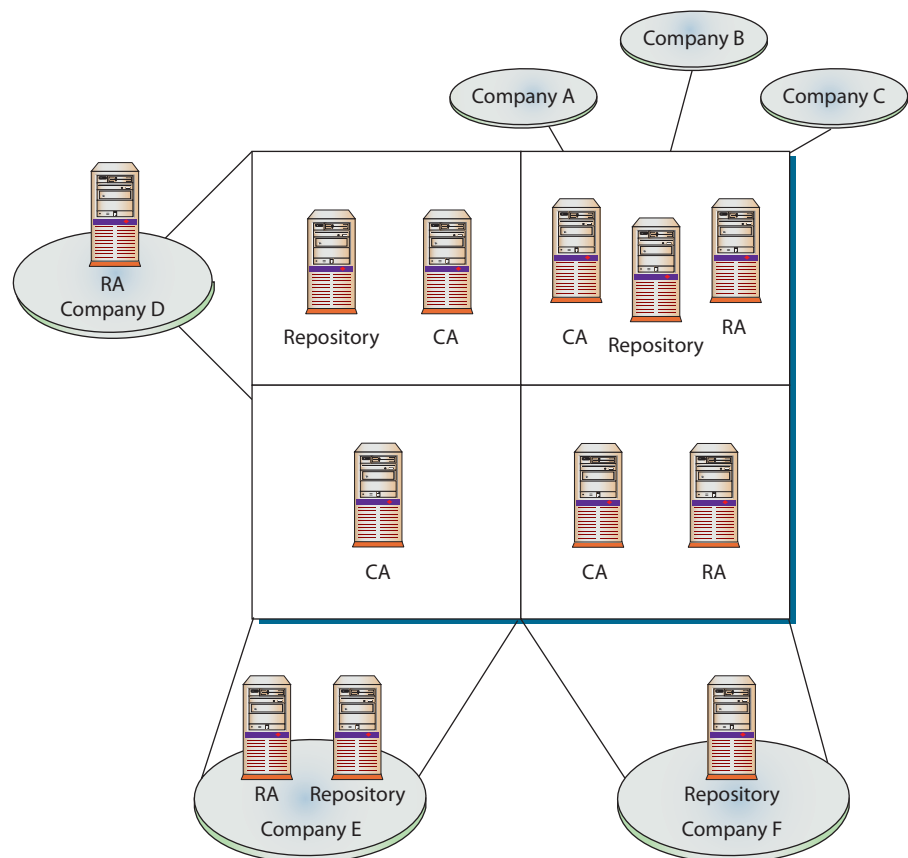
Tech Tip

Outsourced CA vs. Public CA

An outsourced CA is different from a public CA in that it provides dedicated services, and possibly equipment, to an individual company. A public CA, in contrast, can be used by hundreds or thousands of companies—the CA doesn't maintain specific servers and infrastructures for individual companies.

commitment. You need to determine what level of trust the company is willing to give to the service provider and what level of risk it is willing to accept. Often a PKI and its components serve as large security components within a company's enterprise, and allowing a third party to maintain the PKI can introduce too many risks and liabilities that your company is not willing to undertake. The liabilities the service provider is willing to accept, the security precautions and procedures the outsourced CAs provide, and the surrounding legal issues need to be examined before this type of agreement is made.

Some large vertical markets have their own outsourced PKI environments set up because they share similar needs and usually have the same requirements for certification types and uses. This allows several companies within the same market to split the costs of the necessary equipment, and it allows for industry-specific standards to be drawn up and followed. For example, although many medical facilities work differently and have different environments, they have a lot of the same functionality and security needs. If several of them came together, purchased the necessary equipment to provide CA, RA, and CRL functionality, employed one person to maintain it, and then each connected its different sites to the centralized components, the medical facilities could save a lot of money and resources. In this case, not every facility would need to strategically plan its own full PKI, and each would not need to purchase redundant equipment or employ redundant staff members. Figure 7.3 illustrates how one outsourced service provider



• **Figure 7.3** A PKI service provider (represented by the four boxes) can offer different PKI components to companies.

can offer different PKI components and services to different companies, and how companies within one vertical market can share the same resources.

A set of standards can be drawn up about how each different facility should integrate its own infrastructure and how it should integrate with the centralized PKI components. This also allows for less-complicated inter-communication to take place between the different medical facilities, which will ease information-sharing attempts.

Online vs. Offline CA

Certification servers must be online to provide certification services, so why would one have an offline server? The primary reason is security. For a given certificate authority that is only used for periodic functions, such as the signing of specific certificates that are rarely reissued or signed, keeping the server offline except when needed provides a significant level of security to the signing process. Other CA requests, such as CRL and validation requests, can be moved to a validation authority approved by the CA. **Offline CAs** are immune from many forms of attack and serve a valuable function as a backup data reference, while **online CAs** handle normal transactional work.

Stapling

Stapling is the process of combining related items to reduce communication steps. As an example, when someone requests a certificate, the stapling process sends both the certificate and Online Certificate Status Protocol (OCSP) responder information in the same request to avoid the additional fetches the client should perform during path validations.

Pinning

When a certificate is presented for a host, either identifying the host or providing a public key, this information can be saved in an act called **pinning**, which is the process of associating a host with a previously provided X.509 certificate or public key. This can be important for mobile applications that move between networks frequently and are much more likely to be associated with hostile networks where levels of trust are low and the risk of malicious data is high. Pinning assists in security through the avoidance of the use of DNS and its inherent risks when on less-than-secure networks.

The process of reusing a certificate or public key is called *key continuity*. This provides protection from an attacker, provided that the attacker was not in position to attack on the initial pinning. If an attacker is able to intercept and taint the initial contact, then the pinning will preserve the attack. You should pin any time you want to be relatively certain of the remote host's identity, relying on your home network security, and are likely to be operating at a later time in a hostile environment. If you choose to pin, you have two options: you can either pin the certificate or pin the public key.



Tech Tip

Trust Models

Several forms of trust models are associated with certificates. Hierarchical, peer-to-peer, and hybrid are the primary forms, with the web of trust being a form of hybrid. Each of these models has a useful place in the PKI architecture under different circumstances.

■ Trust Models

Potential scenarios exist other than just having more than one CA—each of the companies or each department of an enterprise can actually represent a trust domain itself. A *trust domain* is a construct of systems, personnel, applications, protocols, technologies, and policies that work together to provide a certain level of protection. All of these components can work together seamlessly within the same trust domain because they are known to the other components within the domain and are trusted to some degree. Different trust domains are usually managed by different groups of administrators, have different security policies, and restrict outsiders from privileged access.

Most trust domains (whether individual companies or departments) usually are not islands cut off from the world—they need to communicate with other, less-trusted domains. The trick is to figure out how much two different domains should trust each other as well as how to implement and configure an infrastructure that would allow these two domains to communicate in a way that will not allow security compromises or breaches. This can be more difficult than it sounds.

In the nondigital world, it is difficult to figure out whom to trust, how to carry out legitimate business functions, and how to ensure that one is not being taken advantage of or lied to. Jump into the digital world and add protocols, services, encryption, CAs, RAs, CRLs, and differing technologies and applications, and the business risks can become overwhelming and confusing. So start with a basic question: what criteria will we use to determine whom we trust and to what degree?

One example of trust considered earlier in the chapter is the driver's license issued by the DMV. Suppose, for example, that Bob is buying a lamp from Carol and he wants to pay by check. Since Carol does not know Bob, she does not know if she can trust him or have much faith in his check. But if Bob shows Carol his driver's license, she can compare the name to what appears on the check, and she can choose to accept it. The *trust anchor* (the agreed-upon trusted third party) in this scenario is the DMV because both Carol and Bob trust it more than they trust each other. Bob had to provide documentation to the DMV to prove his identity, and that organization trusted him enough to generate a license. Because Carol trusts the DMV, she decides to trust Bob's check.

Consider another example of a trust anchor. If Joe and Stacy need to communicate through e-mail and would like to use encryption and digital signatures, they will not trust each other's certificate alone. But when each receives the other's certificate and sees that it has been digitally signed by an entity they both do trust—the CA—they have a deeper level of trust in each other. The trust anchor here is the CA. This is easy enough, but when we need to establish trust anchors between different CAs and PKI environments, it gets a little more complicated.


If two companies need to communicate using their individual PKIs, or if two departments within the same company use different CAs, two separate trust domains are involved. The users and devices from these different trust domains need to communicate with each other, and they need to exchange certificates and public keys, which means that trust anchors

need to be identified and a communication channel must be constructed and maintained.

A trust relationship must be established between two issuing authorities (CAs). This happens when one or both of the CAs issue a certificate for the other CA's public key, as shown in Figure 7.4. This means that each CA registers for a certificate and public key from the other CA. Each CA validates the other CA's identification information and generates a certificate containing a public key for that CA to use. This establishes a trust path between the two entities that can then be used when users need to verify other users' certificates that fall within the different trust domains. The trust path can be unidirectional or bidirectional, so either the two CAs trust each other (bidirectional) or only one trusts the other (unidirectional).

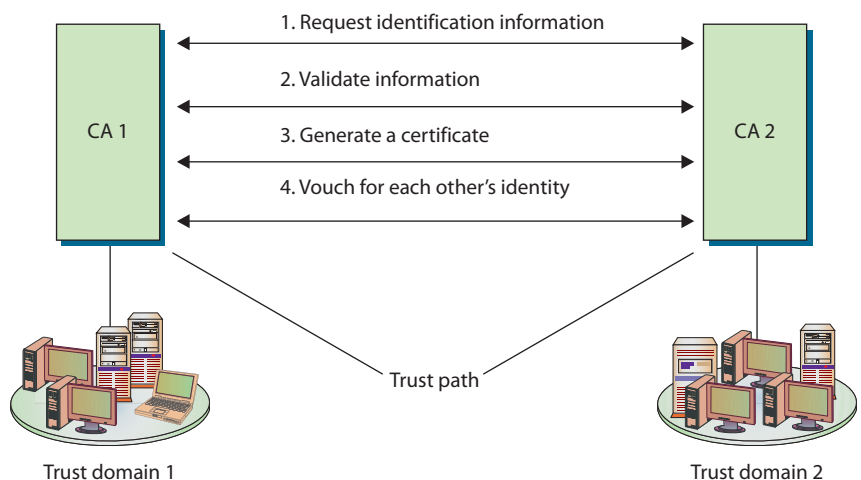
As illustrated in Figure 7.4, all the users and devices in trust domain 1 trust their own CA, CA 1, which is their trust anchor. All users and devices in trust domain 2 have their own trust anchor, CA 2. The two CAs have exchanged certificates and trust each other, but they do not have a common trust anchor between them.

The trust models describe and outline the trust relationships between the different CAs and different environments, which will indicate where the trust paths reside. The trust models and paths need to be thought out before implementation to restrict and control access properly and to ensure that as few trust paths as possible are used. Several different trust models can be used: the hierarchical, peer-to-peer, and hybrid models are discussed later in the chapter.



Three forms of **trust models** are commonly found in PKIs:

- Hierarchical
- Peer-to-peer
- Hybrid

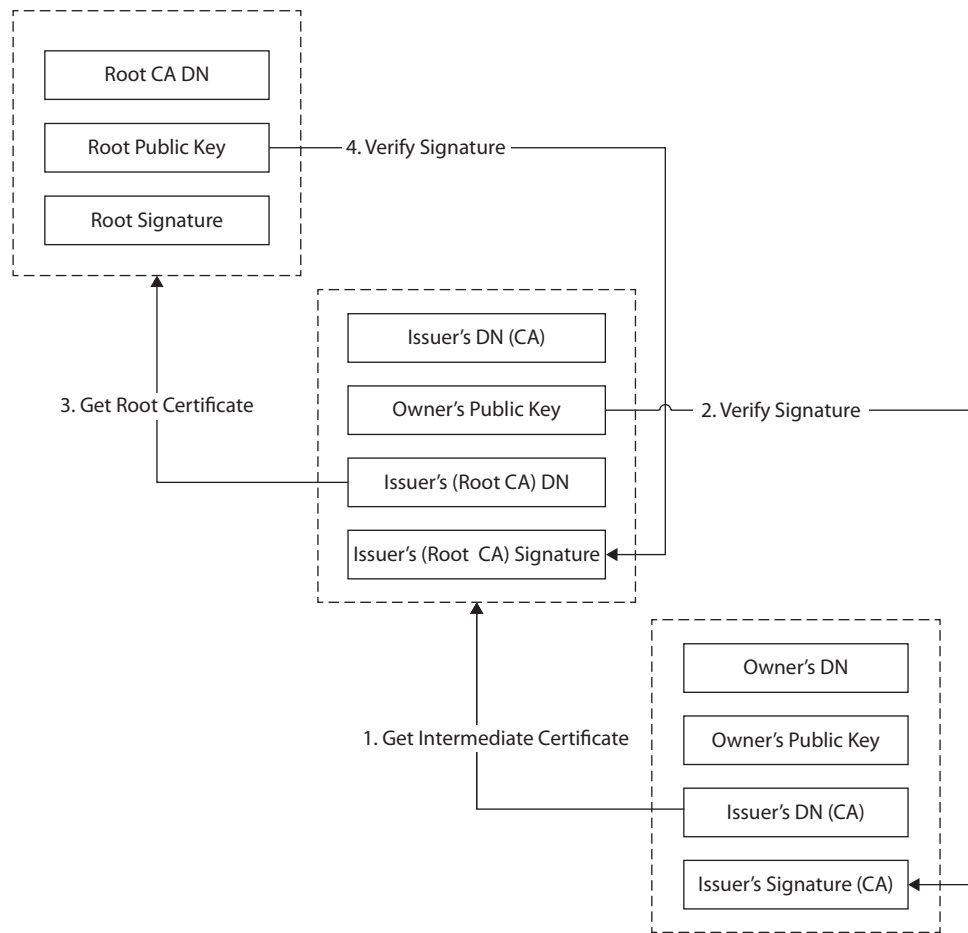


• **Figure 7.4** A trust relationship can be built between two trust domains to set up a communication channel.

Certificate Chaining

Certificates are used to convey identity and public key pairs to users, but this raises the question, why trust the certificate? The answer lies in the certificate chain, a chain of trust from one certificate to another, based on signing by an issuer, until the chain ends with a certificate that the user trusts. This conveys the trust from the trusted certificate to the certificate that is being used. Examining Figure 7.5, we can look at the ordered list of certificates from the one presented to the one that is trusted.

Certificates that sit between the presented certificate and the root certificate are called *chain* or **intermediate certificates**. The intermediate certificate is the signer/issuer of the presented certificate, indicating that it trusts the certificate. The root CA certificate is the signer/issuer of the intermediate certificate, indicating that it trusts the intermediate certificate. The **certificate chaining** is a manner of passing trust down from a trusted root certificate. The chain terminates with a root CA certificate. The root CA certificate is always signed by the CA itself. The signatures of all certificates in the chain must be verified up to the root CA certificate.



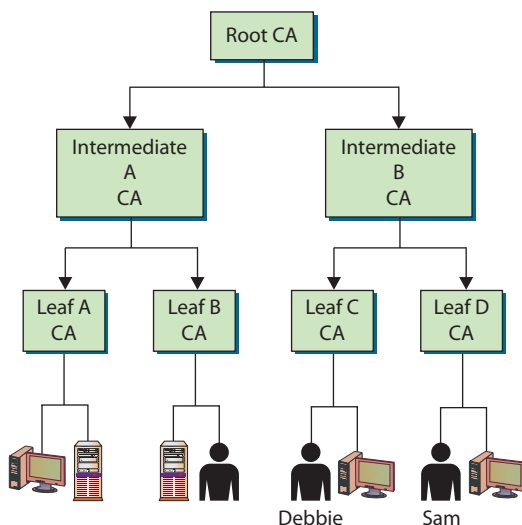
• **Figure 7.5** Certificate chaining

Hierarchical Trust Model

The **hierarchical trust model** is a basic hierarchical structure that contains a root CA, intermediate CAs, leaf CAs, and end-entities. The configuration is that of an inverted tree, as shown in Figure 7.5. The root CA is the ultimate trust anchor for all other entities in this infrastructure, and it generates certificates for the intermediate CAs, which in turn generate certificates for the leaf CAs, and the leaf CAs generate certificates for the end-entities (users, network devices, and applications).

Intermediate CAs function to transfer trust between different CAs. These CAs are referred to as *subordinate CAs* because they are subordinate to the CA they reference. The path of trust is walked up from the subordinate CA to the higher-level CA; in essence, the subordinate CA is using the higher-level CA as a reference.

As shown in Figure 7.6, no bidirectional trusts exist—they are all unidirectional trusts, as indicated by the one-way arrows. Because no other entity can certify and generate certificates for the root CA, it creates a *self-signed certificate*. This means that the



• **Figure 7.6** The hierarchical trust model outlines trust paths.

certificate's Issuer and Subject fields hold the same information, both representing the root CA, and the root CA's public key will be used to verify this certificate when that time comes. This root CA certificate and public key are distributed to all entities within this trust model.

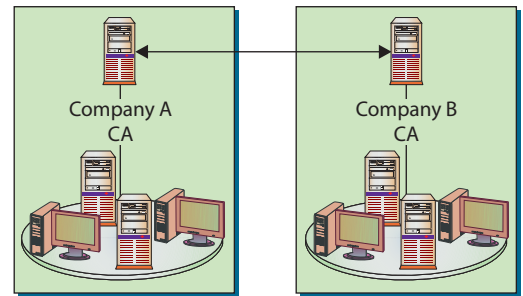
Peer-to-Peer Model

In a **peer-to-peer trust model**, one CA is not subordinate to another CA, and no established trusted anchor between the CAs is involved. The end-entities will look to their issuing CA as their trusted anchor, but the different CAs will not have a common anchor.

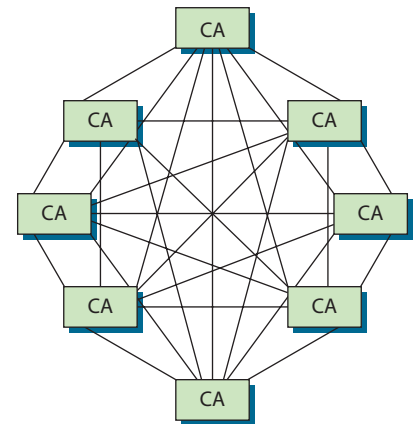
Figure 7.7 illustrates this type of trust model. The two different CAs will certify the public key for each other, which creates a bidirectional trust. This is referred to as *cross-certification* because the CAs are not receiving their certificates and public keys from a superior CA, but instead are creating them for each other.

One of the main drawbacks to this model is scalability. Each CA must certify every other CA that is participating, and a bidirectional trust path must be implemented, as shown in Figure 7.8. If one root CA were certifying all the intermediate CAs, scalability would not be as much of an issue.

Figure 7.8 represents a fully connected *mesh architecture*, meaning that each CA is directly connected to and has a bidirectional trust relationship with every other CA. As you can see in this figure, the complexity of this setup can become overwhelming.



• **Figure 7.7** Cross-certification creates a peer-to-peer PKI model.



• **Figure 7.8** Scalability is a drawback in cross-certification models.

Hybrid Trust Model

A company can be internally complex, and when the need arises to communicate properly with outside partners, suppliers, and customers in an authorized and secured manner, this complexity can make sticking to either the hierarchical or peer-to-peer trust model difficult, if not impossible. In many implementations, the different model types have to be combined to provide the necessary communication lines and levels of trust. In a **hybrid trust model**, the two companies have their own internal hierarchical models and are connected through a peer-to-peer model using cross-certification.

Another option in this hybrid configuration is to implement a bridge CA. Figure 7.9 illustrates the role a bridge CA could play—it is responsible for issuing cross-certificates for all connected CAs and trust domains. The bridge is not considered a root or trust anchor, but merely the entity that generates and maintains the cross-certification for the connected environments.

Walking the Certificate Path

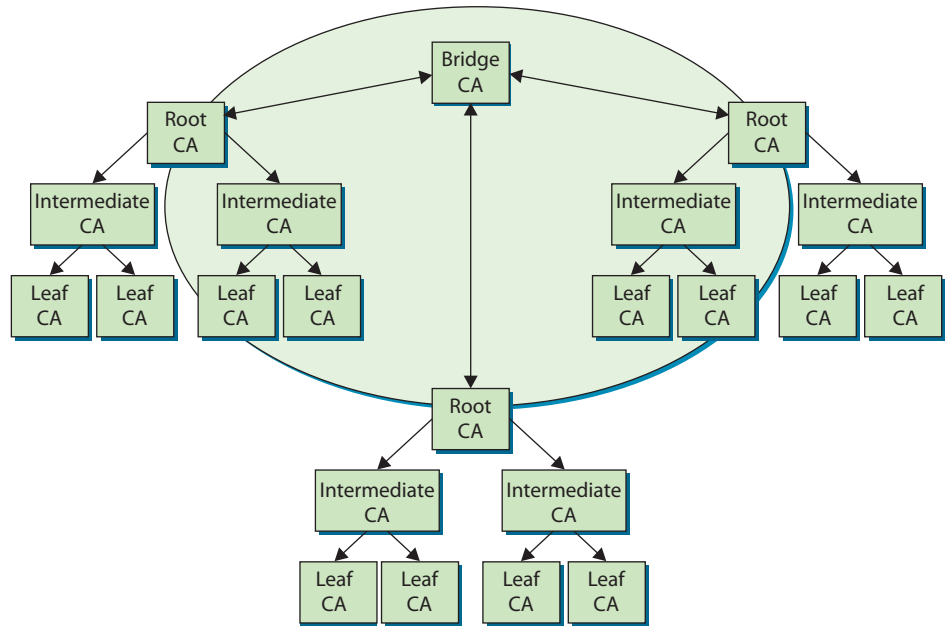
When a user in one trust domain needs to communicate with a user in another trust domain, one user will need to validate the other's certificate. This sounds simple enough, but what it really means is that each certificate for each CA, all the way up to a shared trusted anchor, also must



In any network model, fully connected mesh architectures are wasteful and expensive. In trust-transfer models, the extra level of redundancy is just that: redundant and unnecessary.



Three trust models exist: hierarchical, peer-to-peer, and hybrid. Hierarchical trust is like an upside-down tree, peer-to-peer is a lateral series of references, and hybrid is a combination of hierarchical and peer-to-peer trust.



• **Figure 7.9** A bridge CA can control the cross-certification procedures.



Tech Tip

Root CA

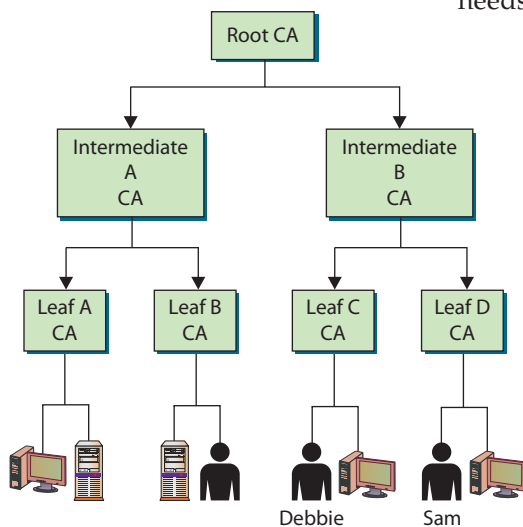
If the root CA's private key were ever compromised, all entities within the hierarchical trust model would be drastically affected, because this is their sole trust anchor. The root CA usually has a small amount of interaction with the intermediate CAs and end-entities, and can therefore be taken offline much of the time. This provides a greater degree of protection for the root CA, because when it is offline it is basically inaccessible.

be validated. If Debbie needs to validate Sam's certificate, as shown in Figure 7.10, she actually also needs to validate the Leaf D CA and Intermediate B CA certificates, as well as Sam's.

So in Figure 7.10, we have a user, Sam, who digitally signs a message and sends it and his certificate to Debbie. Debbie needs to validate this certificate before she can trust Sam's digital signature. Included in Sam's certificate is an Issuer field, which indicates that the certificate was issued by Leaf D CA. Debbie has to obtain Leaf D CA's digital certificate and public key to validate Sam's certificate. Remember that Debbie validates the certificate by verifying its digital signature. The digital signature was created by the certificate issuer using its private key, so Debbie needs to verify the signature using the issuer's public key.

Debbie tracks down Leaf D CA's certificate and public key, but she now needs to verify this CA's certificate, so she looks at the Issuer field, which indicates that Leaf D CA's certificate was issued by Intermediate B CA. Debbie now needs to get Intermediate B CA's certificate and public key.

Debbie's client software tracks this down and sees that the issuer for Intermediate B CA is the root CA, for which she already has a certificate and public key. So Debbie's client software had to follow the **certificate path**, meaning it had to continue to track down and collect certificates until it came upon a self-signed certificate. A self-signed certificate indicates that it was signed by a root CA, and Debbie's software has been configured to trust this entity as her trust anchor, so she can stop there. Figure 7.10 illustrates the steps Debbie's software had to carry out just to be able to verify Sam's certificate.



• **Figure 7.10** Verifying each certificate in a certificate path

This type of simplistic trust model works well within an enterprise that easily follows a hierarchical organizational chart, but many companies cannot use this type of trust model because different departments or offices require their own trust anchors. These demands can be derived from direct business needs or from inter-organizational politics. This hierarchical model might not be possible when two or more companies need to communicate with each other. Neither company will let the other's CA be the root CA, because each does not necessarily trust the other entity to that degree. In these situations, the CAs will need to work in a peer-to-peer relationship instead of in a hierarchical relationship.

■ Digital Certificates

A digital certificate binds an individual's identity to a public key, and it contains all the information a receiver needs to be assured of the identity of the public key owner. After an RA verifies an individual's identity, the CA generates the digital certificate, but how does the CA know what type of data to insert into the certificate?

The certificates are created and formatted based on the **X.509 standard**, which outlines the necessary fields of a certificate and the possible values that can be inserted into the fields. As of this writing, X.509 version 3 is the most current version of the standard. X.509 is a standard of the International Telecommunication Union (www.itu.int). The IETF's Public Key Infrastructure (X.509) working group, or PKIX working group, has adapted the X.509 standard to the more flexible organization of the Internet, as specified in RFC 5280, and is commonly referred to as PKIX for Public Key Infrastructure X.509.

Table 7.1 lists and describes the fields in an X.509 certificate.

Figure 7.11 shows the actual values of the different certificate fields for a particular certificate in Internet Explorer. The version of this certificate is v3 (X.509 v3) and the serial number is also listed—this number is unique for each certificate that is created by a specific CA. The CA used the SHA-1 hashing algorithm to create the message digest value and then signed it using the CA's private key using the RSA algorithm. The actual CA that issued the certificate is Symantec Class 2 Shared Intermediate Certificate Authority, and the valid dates indicate how long this certificate is valid. The subject is Art Conklin, University of Houston, which is the entity that registered this certificate and that is bound to the embedded public key. The actual public key is shown in the lower window and is represented in hexadecimal.

The subject of a certificate is commonly a person, but it does not have to be. The subject can also be a network device (router, web server, firewall, and so on), an application, a department, or a company. Another example used in Table 7.1 is "CN = *.google.com, O = Google LLC, L = Mountain View, S = California, C = US." Each certificate has its own identity that needs to be verified and proven to another entity before secure, trusted communication can be initiated. If a network device is using a certificate for authentication, the certificate may contain the identity of that device. This allows a user of the device to verify its authenticity based on the signed certificate and trust in the signing authority. This trust can be transferred to the identity of the device, indicating authenticity.

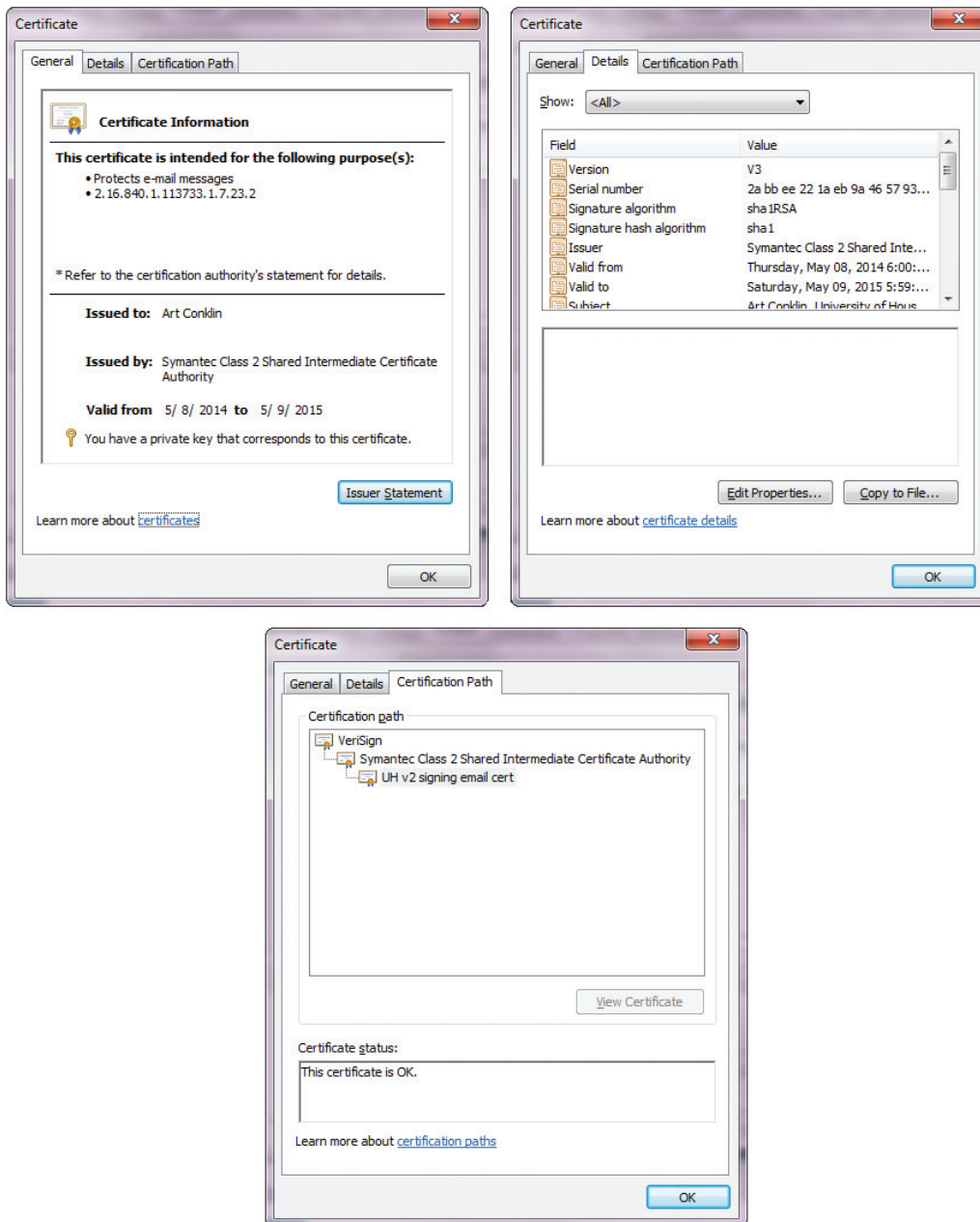
Table 7.1 **X.509 Certificate Fields**

Field Name	Field Description
Certificate Version	X.509 version used for this certificate: Version 1 = 0 Version 2 = 1 Version 3 = 2
Serial Number	A nonnegative integer assigned by the certificate issuer that must be unique to the certificate.
Signature Algorithm Parameters (optional)	The algorithm identifier for the algorithm used by the CA to sign the certificate. The optional Parameters field is used to provide the cryptographic algorithm parameters used in generating the signature.
Issuer	Identification for the entity that signed and issued the certificate. This must be a Distinguished Name within the hierarchy of CAs.
Validity Not valid before time Not valid after time	Specifies a period of time during which the certificate is valid, using a “not valid before” time and a “not valid after” time (expressed in UTC or in a generalized time).
Subject	The Distinguished Name for the certificate owner. This can contain the Common Name and other elements, such as Organization, Location, State and Country: CN = *.google.com, O = Google LLC, L = Mountain View, S = California, C = US.
Subject Public Key Info	An encryption algorithm identifier followed by a bit string for the public key.
Issuer Unique ID	Optional for versions 2 and 3. This is a unique bit-string identifier for the CA that issued the certificate.
Subject Unique ID	Optional for versions 2 and 3. This is a unique bit-string identifier for the subject of the certificate.
Extensions Extension ID Critical Extension Value	Optional for version 3. The extensions area consists of a sequence of extension fields containing an extension identifier, a Boolean field indicating whether the extension is critical, and an octet string representing the value of the extension. Extensions can be defined in standards or defined and registered by organizations or communities.
Thumbprint Algorithm Algorithm Parameters (optional)	Identifies the algorithm used by the CA to sign this certificate. This field must match the algorithm identified in the Signature Algorithm field.
Thumbprint	The signature is the bit-string hash value obtained when the CA signed the certificate. The signature certifies the contents of the certificate, binding the public key to the subject.

Certificate Classes

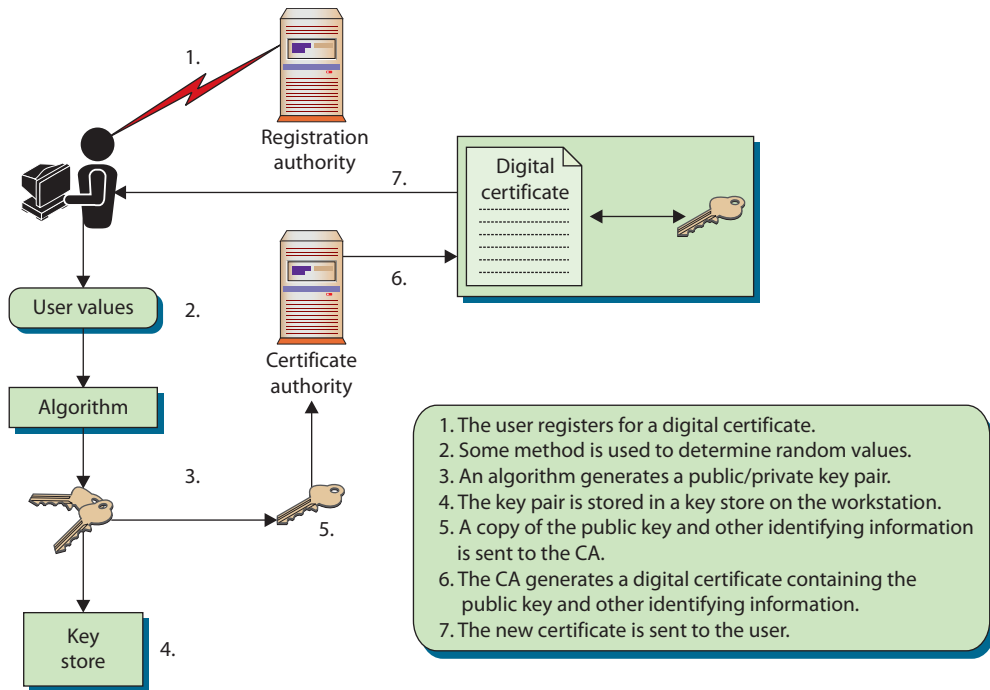
The types of certificates available can vary between different CAs, but usually at least three different types are available, and they are referred to as classes:

- **Class 1** A Class 1 certificate is generally used to verify an individual’s identity through e-mail. A person who receives a Class 1 certificate can use his public/private key pair to digitally sign e-mail and encrypt message contents.
- **Class 2** A Class 2 certificate can be used for software signing. A software vendor would register for this type of certificate so that it could digitally sign its software. This provides integrity for the software after it is developed and released, and it allows the receiver of the software to verify from where the software actually came.
- **Class 3** A Class 3 certificate can be used by a company to set up its own CA, which will allow it to carry out its own identification verification and generate certificates internally.



• **Figure 7.11** Fields within a digital certificate

In most situations, when a user requests a Class 1 certificate, the registration process will require the user to enter specific information into a web-based form. The web page will have a section that accepts the user's public key, or it will step the user through creating a public/private key pair, which will allow the user to choose the size of the keys to be created. Once these steps have been completed, the public key is attached to the certificate registration form and both are forwarded to the RA for processing. The RA is responsible only for the registration process and cannot actually generate a certificate. Once the RA is finished processing the request and verifying the individual's identity, the RA sends the request to the CA. The CA uses the RA-provided information to generate a digital certificate, integrates the



• **Figure 7.12** Steps for obtaining a digital certificate

necessary data into the certificate fields (user identification information, public key, validity dates, proper use for the key and certificate, and so on), and sends a copy of the certificate to the user. These steps are shown in Figure 7.12. The certificate may also be posted to a publicly accessible directory so that others can access it.

Note that a 1:1 correspondence does not necessarily exist between identities and certificates. An entity can have multiple key pairs, using separate public keys for separate purposes. Thus, an entity can have multiple certificates, each attesting to separate public key ownership. It is also possible to have different classes of certificates, again with different keys. This flexibility allows entities total discretion in how they manage their keys, and the PKI manages the complexity by using a unified process that allows key verification through a common interface.

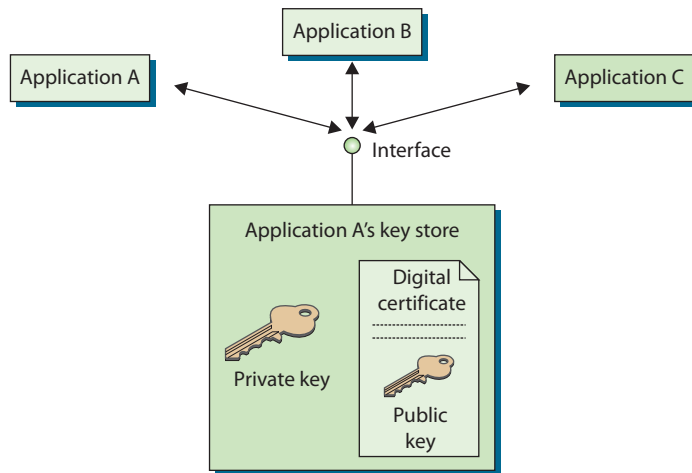
If an application creates a key store that can be accessed by other applications, it will provide a standardized interface, called the *application programming interface (API)*. As an example, Figure 7.13 shows that Application A went through the process of registering a certificate and generating a key pair. It created a key store that provides an interface to allow other applications to communicate with it and use the items held within the store.



The RA verifies the identity of the certificate requestor on behalf of the CA. The CA generates the certificate using information forwarded by the RA.

Certificate Extensions

Certificate extensions allow for further information to be inserted within the certificate, which can be used to provide more functionality in a PKI implementation. Certificate extensions can be standard or private. *Standard certificate extensions* are implemented for every PKI implementation. *Private certificate extensions* are defined for specific organizations (or domains



• **Figure 7.13** Some key stores can be shared by different applications.

within one organization), and they allow companies to further define different, specific uses for digital certificates to best fit their business needs.



Tech Tip

X.509 Digital Certificate Extensions

Following are some key examples of certificate extensions:

- **DigitalSignature** The key used to verify a digital signature.
- **KeyEncipherment** The key used to encrypt other keys used for secure key distribution.
- **DataEncipherment** The key used to encrypt data, which cannot be used to encrypt other keys.
- **CRLSign** The key used to verify a CA signature on a CRL.
- **KeyCertSign** The key used to verify CA signatures on certificates.
- **NonRepudiation** The key used when a nonrepudiation service is being provided.

Several different extensions can be implemented, one being *key usage extensions*, which dictate how the public key that is held within the certificate can be used. Remember that public keys can be used for different functions: symmetric key encryption, data encryption, verifying digital signatures, and more.

A nonrepudiation service can be provided by a third-party notary. In this situation, the sender's digital signature is verified and then signed by the notary so that the sender cannot later deny signing and sending the message. This is basically the same function performed by a traditional notary using paper—validate the sender's identity and validate the time and date of an item being signed and sent. This is required when the receiver needs to be *really* sure of the sender's identity and wants to be legally protected against possible fraud or forgery.



Tech Tip

Critical Flag and Certificate Usage

When an extension is marked as critical, it means that the CA is certifying the key for only that specific purpose. If Joe receives a certificate with a DigitalSignature key usage extension and the critical flag is set, Joe can use the public key only within that certificate to validate digital signatures, and no more. If the extension was marked as noncritical, the key can be used for purposes outside of those listed in the extensions, so in this case it is up to Joe (and his applications) to decide how the key will be used.

If a company needs to be sure that accountable nonrepudiation services will be provided, a trusted time source needs to be used, which can be a trusted third party called a *timestamp authority (TSA)*. Using a trusted time source gives users a higher level of confidence as to *when* specific messages were digitally signed. For example, suppose Barry sends Ron a message and digitally signs it, and Ron later civilly sues Barry over a dispute. This digitally signed message may be submitted by Ron as evidence pertaining to an earlier agreement that Barry now is not fulfilling. If a trusted time source was not used in their PKI environment, Barry could claim that his private key had been compromised before that message was sent. If a trusted time source was implemented, then it could be shown that the message was signed *before* the date on which Barry claims his key was compromised. If a trusted time source is not used, no activity that was carried out within a PKI environment can be truly proven because it is so easy to change system and software time settings.

Critical and Noncritical Extensions

Certificate extensions are considered either *critical* or *noncritical*, which is indicated by a specific flag within the certificate itself. When this flag is set to critical, it means that the extension *must* be understood and processed by the receiver. If the receiver is not configured to understand a particular extension marked as critical, and thus cannot process it properly, the certificate cannot be used for its proposed purpose. If the flag does not indicate that the extension is critical, the certificate can be used for the intended purpose, even if the receiver does not process the appended extension.

Object Identifiers (OID)

Each extension to a certificate has its own ID, expressed as an object identifier, which is a set of values, together with either a critical or noncritical indication. The system using a certificate must reject the certificate if it encounters a critical extension that it does not recognize, or that contains information it cannot process. A noncritical extension may be ignored if it is not recognized but must be processed if it is recognized.

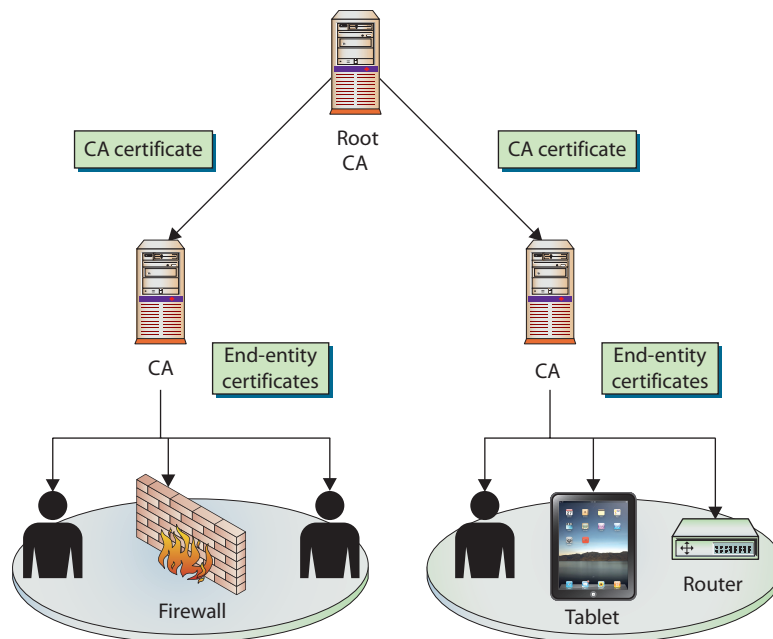
Certificate Attributes

Four main types of certificates are used:

- End-entity certificates
- CA certificates
- Cross-certification certificates
- Policy certificates

End-entity certificates are issued by a CA to a specific subject, such as Joyce, the Accounting department, or a firewall, as illustrated in Figure 7.14. An end-entity certificate is the identity document provided by PKI implementations.

A **CA certificate** can be self-signed, in the case of a standalone or root CA, or it can be issued by a superior CA within a hierarchical model. In the



• **Figure 7.14** End-entity and CA certificates

model in Figure 7.14, the superior CA gives the authority and allows the subordinate CA to accept certificate requests and generate the individual certificates itself. This may be necessary when a company needs to have multiple internal CAs, and different departments within an organization need to have their own CAs servicing their specific end-entities in their sections. In these situations, a representative from each department requiring a CA registers with the higher trusted CA and requests a Certificate Authority certificate. (Public and private CAs are discussed in the “Public Certificate Authorities” and “In-house Certificate Authorities” sections earlier in this chapter, as are the different trust models that are available for companies.)

A **cross-certification certificate**, or *cross-certificate*, is used when independent CAs establish peer-to-peer trust relationships. Simply put, cross-certificates are a mechanism through which one CA can issue a certificate allowing its users to trust another CA.

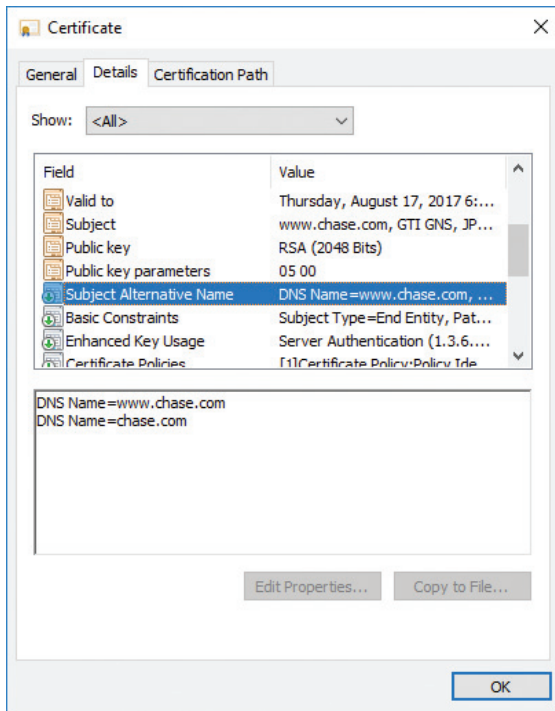
Within sophisticated CAs used for high-security applications, a mechanism is required to provide centrally controlled policy information to PKI clients. This is often done by placing the policy information in a **policy certificate**.

Wildcard

Certificates can be issued to an entity such as example.com. But what if there are multiple entities under example.com that need certificates? In this case, you can either issue distinct certificates for each specific address or use wildcard certificates. **Wildcard certificates** work exactly as one would expect. A certificate issued for *.example.com, for instance, would be valid for both one.example.com and two.example.com.

CN

The **Common Name (CN)** field is represented in the Subject field of the certificate and is the fully qualified domain name (FQDN) for which the



• **Figure 7.15** Subject Alternative Name



SAN certificates allow you to secure a primary domain and then add more domains to the Subject Alternative Name field of the certificate. For example, you can secure all these domains with a single SAN certificate:

- www.example.com
- email.example.com
- intranet.example.com
- www.example.net

certificate is valid. A common representation in the subject line of a certificate may read as follows: CN = *.google.com, O = Google LLC, L = Mountain View, S = California, C = US. The CN is Common Name, O is Organization, L is Location, S is State, and C is Country.

A Distinguished Name (DN) is a term that describes the identifying information in a certificate and is part of the certificate itself. A certificate contains DN information for both the owner or requestor of the certificate (called the Subject DN) and the CA that issues the certificate (called the Issuer DN).

SAN

Subject Alternative Name (SAN) is a field (extension) in a certificate that has several uses. In certificates for machines, it can represent the fully qualified domain name (FQDN) of the machine. For users, it can be the user principal name (UPN), or in the case of an SSL certificate it can indicate multiple domains across which the certificate is valid. Figure 7.15 shows the multiple domains covered by the certificate in the box below the field details. SAN is an extension that is used to a significant degree, as it has become a standard method used in a variety of circumstances.

Code Signing

Certificates can be designated for specific purposes, such as code signing. This is to enable the flexibility of managing certificates for specific functions and reducing the risk in the event of compromise. Code-signing certificates are designated as such in the certificate itself, and the application that uses the certificate adheres to this policy restriction to ensure proper certificate usage.

Self-Signed

Certificates are signed by a higher-level CA, providing a root of trust. As with all chains, there is a final node of trust, the root node. Not all certificates have to have the same root node. A company can create its own certificate chain for use inside the company, and thus it creates its own root node. This certificate must be self-signed, as there is no other “higher” node of trust. What prevents one from signing their own certificates? The trust chain would begin and end with the certificate, and the user would be presented with the dilemma as whether or not to trust the certificate. In the end, all a certificate does is detail a chain of trust to some entity that an end user trusts.

Machine/Computer

Certificates bind identities to keys and provide a means of authentication, which at times is needed for computers. Active Directory Domain Services (AD DS) can keep track of machines in a system via machines identifying themselves using machine certificates. When a user logs in, the system can use either the machine certificate, identifying the machine, or the user certificate, identifying the user—whichever is appropriate for the desired operation.

E-mail

Digital certificates can be used with e-mail systems for items such as digital signatures associated with e-mails. Just as other specialized functions, such as code signing, have their own certificate, it is common for a separate certificate to be used for identity associated with e-mail.

User

User certificates are just that—certificates that identify a user.

Root

Root certificate is the name given to a certificate that forms the initial basis of trust in a trust chain. All certificates are signed by the CA that issues them, and these CAs can be chained together in a trust structure. Following the chain, one climbs the tree of trust until they find a self-signed certificate, indicating it is a root certificate. What determines whether a system trusts a root certificate is whether or not it is in the system's store of trusted certificates. Different vendors such as Microsoft and Apple have trusted root certificate programs where they have determined by corporate policy which CAs they will label as trusted.

Domain Validation

Domain validation is a low-trust means of validation based on control over Transport Layer Security (TLS), where the applicant has been validated by proving some level of control over a Domain Name Service (DNS) domain. This can be automated via checks against a DNS record. A domain validation-based certificate, typically free, offers very little in assurance against a legal entity because the applicant need not supply that information. Domain validation scales well and can be automated with little to no real interaction between an applicant and the CA, but in return it offers little assurance. Domain validation is indicated differently in different browsers, primarily to differentiate it from extended validation certificates.

Extended Validation

Extended validation (EV) certificates are used for HTTPS websites and software to provide a high level of assurance as to the originator's identity. EV certificates use the same methods of encryption to protect certificate integrity as do domain- and organization-validated certificates. The difference in assurance comes from the processes used by a CA to validate an entity's legal identity before issuance. Because of the additional information used during the validation, EV certificates display the legal identity and other legal information as part of the certificate. EV certificates support multiple domains, but do not support wildcards.

To assist users in identifying EV certificates and their enhanced trust, several additional visual clues are provided when EVs are employed. When an EV is implemented in a browser, the legal entity name is displayed, in addition to the URL and a lock symbol, and in most instances the entire URL bar is green. All major browser vendors provide this support, and because the information is included in the certificate itself, this function is web server agnostic.



User certificates are used by users for Encrypting File System (EFS), e-mail, and client authentications, whereas computer certificates help computers to authenticate to the network.



Tech Tip

Certificate Issues

Certificates are a foundational element in establishing trust, and errors in their configuration, implementation, and use can result in improper trust. A certificate that is trusted when it shouldn't be, by whatever means, results in an incorrect assumption of trust. There is no such thing as minor or insignificant issues when it comes to establishing trust. Throughout the text many examples of trust violations can be seen as part of an attack vector.

Certificate Formats

Digital certificates are defined in RFC 5280, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." This RFC describes the X.509 v3 digital certificate format in detail. There are numerous ways to encode the information in a certificate before instantiation as a file, and the different methods result in different file extensions. Common extensions include .der, .pem, .crt, .cer, .pfx, .p12, and .p7b. Although they all can contain certificate information, they are not all directly interchangeable.

DER

Distinguished encoding rules (DER) is one of the Abstract Syntax Notation One (ASN.1) encoding rules that can be used to encode any data object into a binary file. With respect to certificates, the data associated with the certificate, a series of name-value pairs, needs to be converted to a consistent format for digital signing. DER offers a consistent mechanism for this task. A .der file contains binary data and can be used for a single certificate.

PEM

Privacy Enhanced Mail (PEM) is the most common format used by certificate authorities when issuing certificates. PEM comes from RFC 1422, which defined the specification for Privacy Enhanced Mail in 1993 and is a Base64-encoded ASCII file that begins with "-----BEGIN CERTIFICATE-----", followed by the Base64 data, and closes with "-----END CERTIFICATE-----". A .pem file supports multiple digital certificates, including a certificate chain. This file can contain multiple entries, one after another, and can include both public and private keys. Most platforms, however, such as web servers, expect the certificates and private keys to be in separate files. The PEM format for certificate data is used in multiple file types, including .pem, .cer, .crt, and .key files.



If you need to transmit multiple certificates, or a certificate chain, use .pem for encoding. PEM encoding can carry multiple certificates, whereas DER can only carry a single certificate.



The only time CRT and CER can safely be interchanged is when the encoding type is identical (for example, PEM encoded CRT = PEM encoded CER). The latter (.cer) is a file extension for an SSL certificate file format used by web servers to help verify the identity and security of the site in question.

CER

The .cer file extension is used to denote an alternative form, from Microsoft, of .crt files. The .cer/.crt extension is used for certificates and may be encoded as binary DER or as ASCII PEM. The .cer and .crt extensions are nearly synonymous. Again, .cer is most commonly associated with Microsoft Windows systems, whereas .crt is associated with UNIX systems.

KEY

A KEY file (denoted by the file extension .key) can be used both for public and private PKCS#8 keys. The keys may be encoded as binary DER or as ASCII PEM.

PFX

A PKCS#12 file is a portable file format with a .pfx extension. It is a binary format for storing the server certificate, intermediate certificates, and the private key in one file. **Personal Information Exchange (PFX)** files are typically used on Windows machines to import and export certificates and private keys.

P12

P12 is an alternative file extension for a PKCS#12 file format, a binary format for storing the server certificate, intermediate certificates, and the private key in one encrypted file. These files usually have an extension such as .pfx or .p12. They are typically used on Windows machines to import and export certificates and private keys.

P7B

The PKCS#7 or **P7B** format is stored in Base64 ASCII format and has a file extension of .p7b or .p7c. A P7B file begins with “-----BEGIN PKCS7-----” and only contains certificates and chain certificates (intermediate CAs), not the private key. The most common platforms that support P7B files are Microsoft Windows and Java Tomcat.

■ Certificate Lifecycles

Keys and certificates should have lifetime settings that force the user to register for a new certificate after a certain amount of time. Determining the proper length of these lifetimes is a tradeoff: shorter lifetimes limit the ability of attackers to crack them, but longer lifetimes lower system overhead. More-sophisticated PKI implementations perform automated and often transparent key updates to avoid the time and expense of having users register for new certificates when old ones expire.

This means that the certificate and key pair have a lifecycle that must be managed. Certificate management involves administrating and managing each of these phases, including registration, certificate and key generation, renewal, and revocation. Additional management functions include CRL distribution, certificate suspension, and key destruction.



Setting certificate lifetimes way into the future and using them for long periods of time provides attackers with extended windows to attack the cryptography. As stated in Chapter 5, cryptography merely buys time against an attacker; it is never an absolute guarantee.

Registration and Generation

A key pair (public and private keys) can be generated locally by an application and stored in a local key store on the user's workstation. The key pair can also be created by a central key-generation server, which will require secure transmission of the keys to the user. The key pair that is created on the centralized server can be stored on the user's workstation or on the user's smart card, which will allow for more flexibility and mobility.

The act of verifying that an individual indeed has the corresponding private key for a given public key is referred to as *proof of possession*. Not all public/private key pairs can be used for digital signatures, so asking the individual to sign a message and return it to prove that they have the necessary private key will not always work. If a key pair is used for encryption, the RA can send a challenge value to the individual, who, in turn, can use their private key to encrypt that value and return it to the RA. If the RA can successfully decrypt this value with the public key that was provided earlier, the RA can be confident that the individual has the necessary private key and can continue through the rest of the registration phase.

Key regeneration and replacement is usually done to protect against these types of threats, although as the processing power of computers increases



Good key management and proper key replacement intervals protect keys from being compromised through human error. Choosing a large key size makes a brute force attack more difficult.



Tech Tip

Centralized vs. Local Key Generation

In most modern PKI implementations, users have two key pairs. One key pair is often generated by a central server and used for encryption and key transfers. This allows the corporate PKI to retain a copy of the encryption key pair for recovery, if necessary. The second key pair, a digital signature key pair, is usually generated by the user to make sure that they are the only one with a copy of the private key. Nonrepudiation can be challenged if there is any doubt about someone else obtaining a copy of an individual's signature private key. If the key pair was created on a centralized server, that could weaken the case that the individual was the only one who had a copy of their private key. If a copy of a user's signature private key is stored anywhere other than in their possession, or if there is a possibility of someone obtaining the user's key, then true nonrepudiation cannot be provided.

and our knowledge of cryptography and new possible cryptanalysis-based attacks expands, key lifetimes may drastically decrease. As with everything within the security field, it is better to be safe now than to be surprised later and sorry.

The PKI administrator usually configures the minimum required key size that users must use to have a key generated for the first time, and then for each renewal. Most applications provide a drop-down list of possible algorithms to choose from, as well as possible key sizes. The key size should provide the necessary level of security for the current environment. The lifetime of the key should be long enough that continual renewal will not negatively affect productivity, but short enough to ensure that the key cannot be successfully compromised.

CSR

A **certificate signing request (CSR)** is the actual request to a CA containing a public key and the requisite information needed to generate a certificate. The CSR contains all the identifying information that is to be bound to the key by the certificate-generation process.

Renewal

The certificate itself has its own lifetime, which can be different from the key pair's lifetime. The certificate's lifetime is specified by the validity dates inserted into the digital certificate. These are beginning and ending dates indicating the time period during which the certificate is valid. The certificate cannot be used before the start date, and once the end date is met, the certificate is expired and a new certificate will need to be issued.

A renewal process is different from the registration phase in that the RA assumes that the individual has already successfully completed one registration round. If the certificate has not actually been revoked, the original keys and certificate can be used to provide the necessary authentication information and proof of identity for the renewal phase.

The certificate may or may not need to change during the renewal process; this usually depends on why the renewal is taking place. If the certificate just expired and the keys will still be used for the same purpose, a new certificate can be generated with new validity dates. If, however, the key pair functionality needs to be expanded or restricted, new attributes and extensions might need to be integrated into the new certificate. These new functionalities may require more information to be gathered from the individual renewing the certificate, especially if the class changes or the new key uses allow for more powerful abilities.

This renewal process is required when the certificate has fulfilled its lifetime and its end validity date has been met.

Suspension

When the owner of a certificate wishes to mark a certificate as no longer valid prior to its natural expiration, two choices exist: revocation and suspension. Revocation, discussed in the next section, is an action with a permanent

outcome. Instead of being revoked, a certificate can be *suspended*, meaning it is temporarily put on hold. If, for example, Bob is taking an extended vacation and wants to ensure that his certificate will not be compromised or used during that time, he can make a suspension request to the CA. The CRL would list this certificate and its serial number, and in the field that describes why the certificate is revoked, it would instead indicate a hold state. Once Bob returns to work, he can make a request to the CA to remove his certificate from the list.

Another reason to suspend a certificate is if an administrator is suspicious that a private key might have been compromised. While the issue is under investigation, the certificate can be suspended to ensure that it cannot be used.

Certificate Revocation

A certificate can be revoked when its validity needs to be ended before its actual expiration date is met, and this can occur for many reasons: for example, a user may have lost a laptop or a smart card that stored a private key; an improper software implementation may have been uncovered that directly affected the security of a private key; a user may have fallen victim to a social engineering attack and inadvertently given up a private key; data held within the certificate might no longer apply to the specified individual; or perhaps an employee left a company and should not be identified as a member of an in-house PKI any longer. In the last instance, the certificate, which was bound to the user's key pair, identified the user as an employee of the company, and the administrator would want to ensure that the key pair could not be used in the future to validate this person's affiliation with the company. Revoking the certificate does this.

If any of these things happens, a user's private key has been compromised or should no longer be mapped to the owner's identity. A different individual may have access to that user's private key and could use it to impersonate and authenticate as the original user. If the impersonator used the key to digitally sign a message, the receiver would verify the authenticity of the sender by verifying the signature using the original user's public key, and the verification would go through perfectly—the receiver would believe it came from the proper sender and not the impersonator. If receivers could look at a list of certificates that had been revoked before verifying the digital signature, however, they would know not to trust the digital signatures on the list. Because of issues associated with the private key being compromised, revocation is permanent and final—once revoked, a certificate cannot be reinstated. If reinstatement was allowed and a user revoked their certificate, then the unauthorized holder of the private key could use it to restore the certificate validity.

Certificate Revocation List

The CA provides protection against impersonation and similar fraud by maintaining a **certificate revocation list (CRL)**, a list of serial numbers of certificates that have been revoked. The CRL also contains a statement indicating why the individual certificates were revoked and a date when the revocation took place. The list usually contains all certificates that have been revoked within the lifetime of the CA. Certificates that have expired



A certificate suspension can be a useful process tool while investigating whether or not a certificate should be considered valid.



Relying on an expiration date on a certificate to “destroy” the utility of a key will not work. A new certificate can be issued with an “extended date.” To end the use of a key set, an entry in a CRL is the only sure way to prevent reissuance and re-dating of a certificate.



Once revoked, a certificate cannot be reinstated. This is to prevent an unauthorized reinstatement by someone who has unauthorized access to the key(s). A key pair can be reinstated for use by issuing a new certificate if at a later time the keys are found to be secure. The old certificate would still be void, but the new one would be valid.



A certificate should never be assumed to be valid without checking for revocation before each use.



Tech Tip

CRL Reason Codes

Per the X.509 v2 CRL standard, the following reasons are used for revocation:

Reason

Code	Reason
0	Unspecified.
1	All keys compromised; indicates compromise or suspected compromise.
2	CA compromised; used only to revoke CA keys.
3	Affiliation changed; indicates a change of affiliation on the certificate.
4	Superseded; the certificate has been replaced by a more current one.
5	Cessation; the certificate is no longer needed, but no reason exists to suspect it has been compromised.
6	Certificate hold; indicates the certificate will not be issued at this point in time.
7	Remove from CRL; used with a delta CRL to indicate a CRL entry should be removed.

are not the same as those that have been revoked. If a certificate has expired, it means that its end validity date was reached. The format of the CRL message is also defined by X.509. The list is signed, to prevent tampering, and contains information on certificates that have been revoked and the reasons for their revocation. These lists can grow quite long, and as such, there are provisions for date and time stamping the list and for issuing delta lists, which show changes since the last list was issued.

The CA is the entity responsible for the status of the certificates it generates; it needs to be told of a revocation, and it must provide this information to others. The CA is responsible for maintaining the CRL and posting it in a publicly available directory.

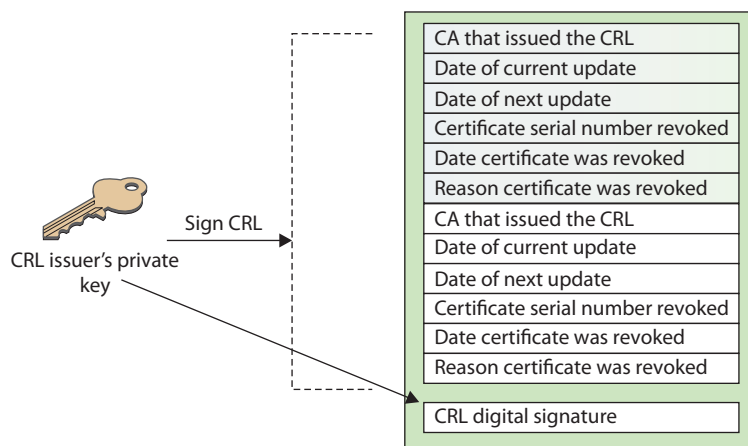
We need to have some system in place to make sure people cannot arbitrarily have others' certificates revoked, whether for revenge or for malicious purposes. When a revocation request is submitted, the individual submitting the request must be authenticated. Otherwise, this could permit a type of denial-of-service attack, in which someone has another person's certificate revoked. The authentication can involve an agreed-upon password that was created during the registration process, but authentication should not be based on the individual proving that they have the corresponding private key, because it may have been stolen, and the CA would be authenticating an imposter.

The CRL's integrity needs to be protected to ensure that attackers cannot modify data pertaining to a revoked certification on the list. If this were allowed to take place, anyone who stole a private key could just delete that key from the CRL and continue to use the private key fraudulently. The integrity of the list also needs to be protected to ensure that bogus data is not added to it. Otherwise, anyone could add another person's certificate to the list and effectively revoke that person's certificate. The only entity that should be able to modify any information on the CRL is the CA.

The mechanism used to protect the integrity of a CRL is a *digital signature*. The CA's revocation service creates a digital signature for the CRL, as shown in Figure 7.16. To validate a certificate, the user accesses the directory where the CRL is posted, downloads the list, and verifies the CA's digital signature to ensure that the proper authority signed the list and to ensure that the list was not modified in an unauthorized manner. The user



The certificate revocation list is an essential item to ensure a certificate is still valid. CAs post CRLs in publicly available directories to permit automated checking of certificates against the list before certificate use by a client. A user should never trust a certificate that has not been checked against the appropriate CRL.



• **Figure 7.16** The CA digitally signs the CRL to protect its integrity.

then looks through the list to determine whether the serial number of the certificate they are trying to validate is listed. If the serial number is on the list, the private key should no longer be trusted, and the public key should no longer be used. This can be a cumbersome process, so it has been automated in several ways, which are described in the next section.

One concern is how up to date the CRL is—how often is it updated and does it actually reflect *all* the certificates currently revoked? The actual frequency with which the list is updated depends on the CA and its certification practices statement (CPS). It is important that the list is updated in a timely manner so that anyone using the list has the most current information.

CRL Distribution

CRL files can be requested by individuals who need to verify and validate a newly received certificate, or the files can be periodically pushed down (sent) to all users participating within a specific PKI. This means the CRL can be pulled (downloaded) by individual users when needed or pushed down to all users within the PKI on a timed interval.

The actual CRL file can grow substantially and transmitting this file and requiring PKI client software on each workstation to save and maintain it can use a lot of resources. Therefore, the smaller the CRL is, the better. It is also possible to first push down the full CRL and subsequently push down only *delta* CRLs, which contain only the changes to the original or base CRL. This can greatly reduce the amount of bandwidth consumed when CRLs are updated.

In implementations where the CRLs are not pushed down to individual systems, the users' PKI software needs to know where to look for the posted CRL that relates to the certificate it is trying to validate. The certificate might have an extension that points the validating user to the necessary *CRL distribution point*. The network administrator sets up the distribution points, and one or more points can exist for a particular PKI. The distribution point holds one or more lists containing the serial numbers of revoked certificates, and the user's PKI software scans the list(s) for the serial number of the certificate the user is attempting to validate. If the serial number is not present, the user is assured that it has not been revoked. This approach helps point users to the right resource and also reduces the amount of information that needs to be scanned when checking that a certificate has not been revoked.

Online Certificate Status Protocol (OCSP)

One last option for checking distributed CRLs is an *online service*. When a client user needs to validate a certificate and ensure that it has not been revoked, they can communicate with an online service that will query the necessary CRLs available within the environment. This service can query the lists for the client instead of pushing down the full CRL to each and every system. So if Joe receives a certificate from Stacy, he can contact an online service and send to it the serial number listed in the certificate Stacy sent. The online service would query the necessary CRLs and respond to Joe, indicating whether or not that serial number was listed as being revoked.

One of the protocols used for online revocation services is the **Online Certificate Status Protocol (OCSP)**, a request and response protocol that obtains the serial number of the certificate that is being validated and



Tech Tip

Authority Revocation

Lists

*In some PKI implementations, a separate revocation list is maintained for CA keys that have been compromised or should no longer be trusted. This list is known as an **authority revocation list (ARL)**. In the event that a CA's private key is compromised or a cross-certification is cancelled, the relevant certificate's serial number is included in the ARL. A client can review an ARL to make sure the CA's public key can still be trusted.*



Certificate revocation checks are done either by examining the CRL or by using OCSP to see if a certificate has been revoked.



Tech Tip

Historical Retention of Certificates

Note that in modern PKIs, encryption key pairs usually must be retained long after they expire so that users can decrypt information that was encrypted with the old keys. For example, if Bob encrypts a document using his current key and the keys are updated three months later, Bob's software must maintain a copy of the old key so he can still decrypt the document. In the PKI world, this issue is referred to as key history maintenance.

reviews revocation lists for the client. The protocol has a responder service that reports the status of the certificate back to the client, indicating whether it has been revoked, is valid, or has an unknown status. This protocol and service saves the client from having to find, download, and process the right lists.

The goal is to make sure that no one can gain access to a key after its lifetime has ended and use that key for malicious purposes. An attacker might use the key to digitally sign or encrypt a message with the hopes of tricking someone else about their identity (this would be an example of a man-in-the-middle attack). Also, if the attacker is performing some type of brute force attack on your cryptosystem, trying to figure out specific keys that were used for encryption processes, obtaining an old key could give the attacker more insight into how your cryptosystem generates keys. The less information you supply to potential hackers, the better.

Key Destruction

Key pairs and certificates have set *lifetimes*, meaning that they will expire at some specified time. It is important that the certificates and keys are properly destroyed when that time comes, wherever the keys are stored (on users' workstations, centralized key servers, USB token devices, smart cards, and so on).

■ Certificate Repositories

Once the requestor's identity has been proven, a certificate is registered with the public side of the key pair provided by the requestor. Public keys must be available to anybody who requires them to communicate within a PKI environment. These keys, and their corresponding certificates, are usually held in a publicly available repository. **Certificate repository** is a general term that describes a centralized directory that can be accessed by a subset of individuals. The directories are usually LDAP-compliant, meaning that they can be accessed and searched via a Lightweight Directory Access Protocol (LDAP) query from an LDAP client.

When an individual initializes communication with another, the sender can send their certificate and public key to the receiver, which will allow the receiver to communicate with the sender using encryption or digital signatures (or both) without needing to track down the necessary public key in a certificate repository. This is equivalent to the sender saying, "If you would like to encrypt any future messages you send to me, or if you would like the ability to verify my digital signature, here are the necessary components." But if a person wants to encrypt the first message sent to the receiver, the sender needs to find the receiver's public key in a certificate repository.

A certificate repository is a holding place for individuals' certificates and public keys that are participating in a particular PKI environment. The security requirements for repositories themselves are not as high as those needed for actual CAs and for the equipment and software used to carry out CA functions. Since each certificate is digitally signed by the CA, if a

certificate stored in the certificate repository is modified, the recipient will be able to detect this change and know not to accept the certificate as valid.

Sharing Key Stores

Different applications from the same vendor may share key stores. Microsoft applications keep user keys and certificates in a Registry entry within that user's profile. The applications can then save and retrieve them from this single location or key store. Other applications could also use the same keys if they knew where they were stored by using Registry API calls.

The local key store is just one location where these items can be held. Often, the digital certificate and public key are also stored in a certificate repository (as discussed earlier in the "Certificate Repositories" section of this chapter) so that they are available to a subset of individuals.

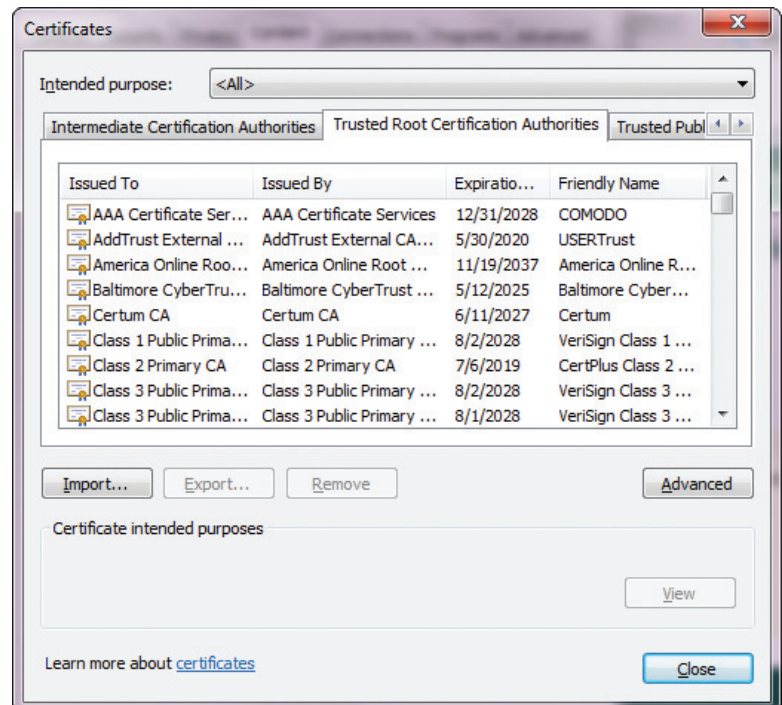
Trust and Certificate Verification

We need to use a PKI if we do not automatically trust individuals we do not know. Security is about being suspicious and being safe, so we need a third party that we *do* trust to vouch for the other individual before confidence can be instilled and sensitive communication can take place. But what does it mean that we trust a CA, and how can we use this to our advantage?

When a user chooses to trust a CA, they will download that CA's digital certificate and public key, which will be stored on their local computer. Most browsers have a list of CAs configured to be trusted by default, so when a user installs a new web browser, several of the most well-known and most trusted CAs will be trusted without any change of settings. An example of this listing is shown in Figure 7.17.

In the Microsoft CAPI (Cryptographic Application Programming Interface) environment, the user can add and remove CAs from this list as needed. In production environments that require a higher degree of protection, this list will be pruned, and possibly the only CAs listed will be the company's *internal* CAs. This ensures that digitally signed software will be automatically installed only if it was signed by the company's CA. Other products, such as Entrust, use centrally controlled policies to determine which CAs are to be trusted, instead of expecting the user to make these critical decisions.

A number of steps are involved in checking the validity of a message. Suppose, for example, that Maynard receives a digitally signed message from Joyce, whom he does not know or trust. Joyce has also included her digital certificate with her message, which has her public key embedded



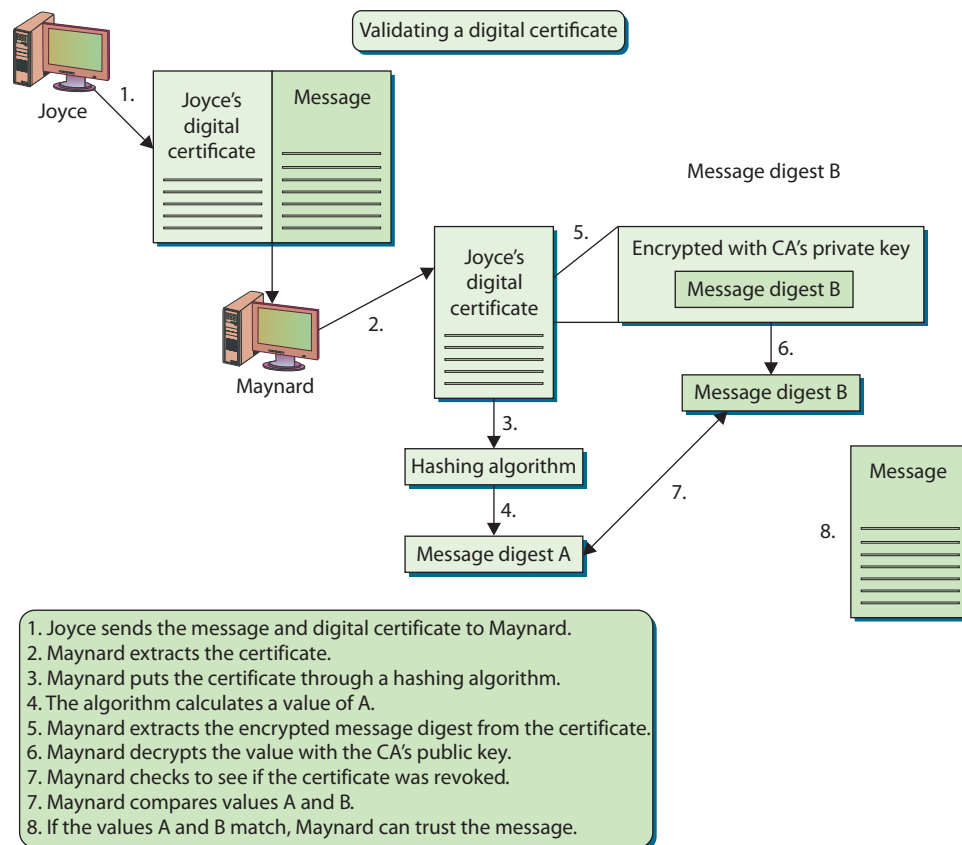
• **Figure 7.17** Browsers have a long list of CAs configured to be trusted by default.



Tech Tip

Distinguished Names

A *Distinguished Name* is a label that follows the X.500 standard. This standard defines a naming convention that can be employed so that each subject within an organization has a unique name. An example is {Country = US, Organization = Real Secure, Organizational Unit = R&D, Location = Washington}. CAs use Distinguished Names to identify the owners of specific certificates.



• **Figure 7.18** Steps for verifying the authenticity and integrity of a certificate



Because certificates produce chains of trust, having an unnecessary certificate in your certificate store could lead to trust problems. Best practices indicate that you should understand the certificates in your store as well as the need for each. When in doubt, remove it. If the certificate is needed, you can add it back later.

within it. Before Maynard can be sure of the authenticity of this message, he has some work to do. The steps are illustrated in Figure 7.18.

First, Maynard sees which CA signed Joyce's certificate and compares it to the list of CAs he has configured within his computer. He trusts the CAs in his list and no others. (If the certificate was signed by a CA that he does not have in the list, he would not accept the certificate as being valid, and thus he could not be sure that this message was actually sent from Joyce or that the attached key was actually her public key.)

Maynard sees that the CA that signed Joyce's certificate is indeed in his list of trusted CAs, so he now needs to verify that the certificate has not been altered. Using the CA's public key and the digest of the certificate, Maynard can verify the integrity of the certificate. Then Maynard can be assured that this CA did actually create the certificate, so he can now trust the origin of Joyce's certificate. The use of digital signatures allows certificates to be saved in public directories without concern for them being accidentally or intentionally altered. If a user extracts a certificate from a repository and creates a message digest value that does not match the digital signature embedded within the certificate itself, that user will know that the certificate has been modified by someone other than the CA and will know not to accept the validity of the corresponding public key. Similarly, an attacker could not create a new message digest, encrypt it, and embed it within the certificate because they would not have access to the CA's private key.

But Maynard is not done yet. He needs to be sure that the issuing CA has not revoked this certificate. The certificate also has start and stop dates, indicating a time during which the certificate is valid. If the start date hasn't happened yet or the stop date has been passed, the certificate is not valid. Maynard reviews these dates to make sure the certificate is still deemed valid.

Another step Maynard may go through is to check whether this certificate has been revoked for any reason. To do so, he will refer to the *certificate revocation list (CRL)* to see if Joyce's certificate is listed. He could check the CRL directly with the CA that issued the certificate, or he could use a specialized online service that supports the Online Certificate Status Protocol (OCSP). (Certificate revocation and list distribution were explained in the "Certificate Lifecycles" section, earlier in this chapter.)

Maynard now trusts that this certificate is legitimate and that it belongs to Joyce. Now what does he need to do? The certificate holds Joyce's public key, which he needs to validate the digital signature she appended to her message, so Maynard extracts Joyce's public key from her certificate, runs her message through a hashing algorithm, and calculates a message digest value of X. He then uses Joyce's public key to decrypt her digital signature (remember that a digital signature is just a message digest encrypted with a private key). This decryption process provides him with another message digest of value Y. Maynard compares values X and Y, and if they are the same, he is assured that the message has not been modified during transmission. Thus, he has confidence in the integrity of the message. But how does Maynard know that the message actually came from Joyce? Because he can decrypt the digital signature using her public key, which indicates that only the associated private key could have been used. There is a miniscule risk that someone could create an identical key pair, but given the enormous keyspace for public keys, this is impractical. The public key can only decrypt something that was encrypted with the related private key, and only the owner of the private key is supposed to have access to it. Maynard can be sure that this message came from Joyce.

After all of this, he reads her message, which says, "Hi. How are you?" All of that work just for this message? Maynard's blood pressure would surely go through the roof if he had to do all of this work only to end up with a short and not very useful message. Fortunately, all of this PKI work is performed without user intervention and happens behind the scenes. Maynard didn't have to exert any energy. He simply replies, "Fine. How are you?"



Tech Tip

Validating a Certificate

The following steps are required for validating a certificate:

1. Compare the CA that digitally signed the certificate to a list of CAs that have already been loaded into the receiver's computer.
2. Calculate a message digest for the certificate.
3. Use the CA's public key to decrypt the digital signature and recover what is claimed to be the original message digest embedded within the certificate (validating the digital signature).
4. Compare the two resulting message digest values to ensure the integrity of the certificate.
5. Review the identification information within the certificate, such as the e-mail address.
6. Review the validity dates.
7. Check a revocation list to see if the certificate has been revoked.

■ Centralized and Decentralized Infrastructures

Keys used for authentication and encryption within a PKI environment can be generated in a centralized or decentralized manner. In a *decentralized* approach, software on individual computers generates and stores cryptographic keys local to the systems themselves. In a *centralized* infrastructure, the keys are generated and stored on a central server, and the

keys are transmitted to the individual systems as needed. You might choose one type over the other for several reasons.

If a company uses an asymmetric algorithm that is resource-intensive to generate the public/private key pair, and if large (and resource-intensive) key sizes are needed, then the individual computers might not have the necessary processing power to produce the keys in an acceptable fashion. In this situation, the company can choose a centralized approach in which a very high-end server with powerful processing capabilities is used, probably along with a hardware-based random number generator.

Central key generation and storage offer other benefits as well. For example, it is much easier to back up the keys and implement key recovery procedures with central storage than with a decentralized approach. Implementing a key recovery procedure on each and every computer holding one or more key pairs is difficult, and many applications that generate their own key pairs do not usually interface well with a centralized archive system. This means that if a company chooses to allow its individual users to create and maintain their own key pairs on their separate workstations, no real key recovery procedure can be put in place. This puts the company at risk. If an employee leaves the organization or is unavailable for one reason or another, the company may not be able to access its own business information that was encrypted by that employee.

So a centralized approach seems like the best approach, right? Well, the centralized method has some drawbacks to consider, too. Secure key distribution is a tricky event. This can be more difficult than it sounds. A technology needs to be employed that will send the keys in an encrypted manner, ensure the keys' integrity, and make sure that only the intended users are receiving the keys.

Also, the server that centrally stores the keys needs to be highly available and is a potential single point of failure, so some type of fault tolerance or redundancy mechanism might need to be put into place. If that one server goes down, users will not be able to access their keys, which might prevent them from properly authenticating to the network, resources, and applications. Also, since all the keys are in one place, the server is a prime target for an attacker—if the central key server is compromised, the whole environment is compromised.

One other issue pertains to how the keys will actually be used. If a public/private key pair is being generated for digital signatures, and if the company wants to ensure that this key pair can be used to provide *true* authenticity and nonrepudiation, the keys should not be generated at a centralized server. This would introduce doubt that only the one person had access to a specific private key. It is better to generate end-user keys on a local machine to eliminate doubt about who did the work and “owns” the keys.

If a company uses smart cards to hold users' private keys, each private key often has to be generated on the card itself and cannot be copied for archiving purposes. This is a disadvantage of the centralized approach. In addition, some types of applications have been developed to create their own public/private key pairs and do not allow other keys to be imported and used. This means the keys would have to be created locally by these applications, and keys from a central server could not be used. These are just some of the considerations that need to be evaluated before any decision is made and implementation begins.

Hardware Security Modules

PKIs can be constructed in software without special cryptographic hardware, and this is perfectly suitable for many environments. But software can be vulnerable to viruses, hackers, and hacking. If a company requires a higher level of protection than a purely software-based solution can provide, several hardware-based solutions are available. A **hardware security module (HSM)** is a physical device that safeguards cryptographic keys. HSMs enable a higher level of security for the use of keys, including generation and authentication.

In most situations, HSM solutions are used only for the most critical and sensitive keys, which are the root key and possibly the intermediate CA private keys. If those keys are compromised, the whole security of the PKI is gravely threatened. If a person obtained a root CA private key, they could digitally sign any certificate, and that certificate would be quickly accepted by all entities within the environment. Such an attacker might be able to create a certificate that has extremely high privileges, perhaps allowing them to modify bank account information in a financial institution, and no alerts or warnings would be initiated because the ultimate CA, the root CA, signed it.

Private Key Protection

Although a PKI implementation can be complex, with many different components and options, a critical concept common to all PKIs must be understood and enforced: the private key needs to stay private. A digital signature is created solely for the purpose of proving who sent a particular message by using a private key. This rests on the assumption that only one person has access to this private key. If an imposter obtains a user's private key, authenticity and nonrepudiation can no longer be claimed or proven.

When a private key is generated for the first time, it must be stored somewhere for future use. This storage area is referred to as a *key store*, and it is usually created by the application registering for a certificate, such as a web browser, smart card software, or other application. In most implementations, the application will prompt the user for a password, which will be used to create an encryption key that protects the key store. So, for example, if Cheryl used her web browser to register for a certificate, her private key would be generated and stored in the key store. Cheryl would then be prompted for a password, which the software would use to create a key that will encrypt the key store. When Cheryl needs to access this private key later that day, she will be prompted for the same password, which will decrypt the key store and allow her access to her private key.

Unfortunately, many applications do not require that a strong password be created to protect the key store, and in some implementations the user can choose not to provide a password at all. The user still has a private key available, and it is bound to the user's identity, so why is a password even necessary? If, for example, Cheryl decided not to use a password, and another person sat down at her computer, he could use her web browser and her private key and digitally sign a message that contains a nasty virus. If Cheryl's coworker Cliff received this message, he would think it came from Cheryl, open the message, and download the virus. The moral to this



Tech Tip

Storing Critical Keys

HSMs take many different forms, including embedded cards, network-attached devices, and even USB flash drives. HSMs assist in the use of cryptographic keys across the lifecycle. They can provide dedicated support for centralized lifecycle management, from generation to distribution, storage, termination, archiving, and recordkeeping. HSMs can increase the efficiency of cryptographic operations and assist in compliance efforts. Common uses include use in PCI DSS solutions, DNSSEC, signing operations (including certificates, code, documents, and e-mail), and large-scale data encryption efforts.



The security associated with the use of public key cryptography revolves around the security of the private key. Nonrepudiation depends on the principle that the private key is only accessible to the holder of the key. If another person has access to the private key, they can impersonate the proper key holder.

story is that users should be required to provide some type of authentication information (password, smart card, PIN, or the like) before being able to use private keys. Otherwise, the keys could be used by other individuals or imposters, and authentication and nonrepudiation would be of no use.

Because a private key is a crucial component of any PKI implementation, the key itself should contain the necessary characteristics and be protected at each stage of its life. The following list sums up the characteristics and requirements of proper private key use:

- The key size should provide the necessary level of protection for the environment.
- The lifetime of the key should correspond with how often it is used and the sensitivity of the data it is protecting.
- The key should be changed at the end of its lifetime and not used past its allowed lifetime.
- Where appropriate, the key should be properly destroyed at the end of its lifetime.
- The key should never be exposed in clear text.
- No copies of the private key should be made if it is being used for digital signatures.
- The key should not be shared.
- The key should be stored securely.
- Authentication should be required before the key can be used.
- The key should be transported securely.
- Software implementations that store and use the key should be evaluated to ensure they provide the necessary level of protection.

If digital signatures will be used for legal purposes, these points and others may need to be audited to ensure that true authenticity and nonrepudiation are provided.



The most sensitive and critical public/private key pairs are those used by CAs to digitally sign certificates. These need to be highly protected because if they were ever compromised, the trust relationship between the CA and all of the end-entities would be threatened. In high-security environments, these keys are often kept in a tamperproof hardware encryption store, such as an HSM, and are accessible only to individuals with a need to know.

Key Recovery

One individual could have one, two, or many key pairs that are tied to their identity. That is because users may have different needs and requirements for public/private key pairs. As mentioned earlier, certificates can have specific attributes and usage requirements dictating how their corresponding keys can and cannot be used. For example, David can have one key pair he uses to encrypt and transmit symmetric keys, another key pair that allows him to encrypt data, and yet another key pair to perform digital signatures. David can also have a digital signature key pair for his work-related activities and another key pair for personal activities, such as e-mailing his friends. These key pairs need to be used only for their intended purposes, and this is enforced through certificate attributes and usage values.

If a company is going to perform key recovery and maintain a key-recovery system, it will generally back up only the key pair used to encrypt data, not the key pairs that are used to generate digital signatures. The reason that a company archives keys is to ensure that if a person leaves the company, falls off a cliff, or for some reason is unavailable to decrypt

important company information, the company can still get to its company-owned data. This is just a matter of the organization protecting itself. A company would not need to be able to recover a key pair that is used for digital signatures, since those keys are to be used only to prove the authenticity of the individual who sent a message. A company would not benefit from having access to those keys and really should not have access to them because they are tied to one individual for a specific purpose.

Two systems are important for backing up and restoring cryptographic keys: key archiving and key recovery. **Key archiving** is a way of backing up keys and securely storing them in a repository; **key recovery** is the process of restoring lost keys to the users or the company.

If keys are backed up and stored in a centralized computer, this system must be tightly controlled, because if it were compromised, an attacker would have access to all keys for the entire infrastructure. Also, it is usually unwise to authorize a single person to be able to recover all the keys within the environment, because that person could use this power for evil purposes instead of just recovering keys when they are needed for legitimate purposes. In security systems, it is best not to fully trust anyone.

Dual control can be used as part of a system to back up and archive data encryption keys. PKI systems can be configured to require multiple individuals to be involved in any key recovery process. When key recovery is required, at least two people can be required to authenticate using the key recovery software before the recovery procedure is performed. This enforces *separation of duties*, which means that one person cannot complete a critical task alone. Requiring two individuals to recover a lost key together is called **dual control**, which simply means that two people have to be present to carry out a specific task.

This approach to key recovery is referred to as “ m of n authentication,” where n number of people can be involved in the key recovery process, but at least m (which is a smaller number than n) *must* be involved before the task can be completed. The goal is to minimize fraudulent or improper use of access and permissions. A company would not require all possible individuals to be involved in the recovery process, because getting all the people together at the same time could be impossible considering meetings, vacations, sick time, and travel. At least some of all possible individuals must be available to participate, and this is the subset m of the number n . This form of secret splitting can increase security by requiring multiple people to perform a specific function. Requiring too many people for the m subset increases issues associated with availability, whereas requiring too few increases the risk of a small number of people colluding to compromise a secret.

All key recovery procedures should be highly audited. The audit logs should capture at least what keys were recovered, who was involved in the process, and the time and date. Keys are an integral piece of any encryption cryptosystem and are critical to a PKI environment, so you need to track who does what with them.

Key Escrow

Key recovery and *key escrow* are terms that are often used interchangeably, but they actually describe two different things. You should not use them interchangeably after you have read this section.



Key archiving is the process of storing a set of keys to be used as a backup should something happen to the original set. *Key recovery* is the process of using the backup keys.



Recovery agent is the term for an entity that is given a public key certificate for recovering user data that is encrypted. This is the most common type of recovery policy used in PKI, but it adds the risk of the recovery agent having access to secured information.



Tech Tip

Keysplitting

Secret splitting using m of n authentication schemes can improve security by requiring that multiple people perform critical functions, preventing a single party from compromising a secret.



Key recovery is a process that allows for lost keys to be recovered. *Key escrow* is a process of giving keys to a third party so that they can decrypt and read sensitive information when this need arises.



Key escrow, allowing another trusted party to hold a copy of a key, has long been a controversial topic. This essential business process provides continuity should the authorized key-holding party leave an organization without disclosing keys. The security of the escrowed key is a concern, and it needs to be managed at the same security level as for the original key.

Key escrow is the process of giving keys to a third party so that they can decrypt and read sensitive information if the need arises. Key escrow almost always pertains to handing over encryption keys to the government, or to another higher authority, so that the keys can be used to collect evidence during investigations. A key pair used in a person's place of work may be required to be escrowed by the employer for two reasons. First, the keys are property of the company, issued to the worker for use. Second, the company may have need for them after an employee leaves the firm.

Several movements, supported by parts of the U.S. government, would require all or many people residing in the United States to hand over copies of the keys they use to encrypt communication channels. The movement in the late 1990s behind the Clipper chip is the most well-known effort to implement this requirement and procedure. It was suggested that all American-made communication devices should have a hardware encryption chip within them. The chip could be used to encrypt data going back and forth between two individuals, but if a government agency decided that it should be able to eavesdrop on this dialogue, it would just need to obtain a court order. If the court order was approved, a law enforcement agent would take the order to two escrow agencies, each of which would have a piece of the key that was necessary to decrypt this communication information. The agent would obtain both pieces of the key and combine them, which would allow the agent to listen in on the encrypted communication outlined in the court order.

The Clipper chip standard never saw the light of day because it seemed too "Big Brother" to many American citizens. But the idea was that the encryption keys would be escrowed to two agencies, meaning that each agency would hold one piece of the key. One agency could not hold the whole key, because it could then use this key to wiretap people's conversations illegally. Splitting up the key is an example of separation of duties, put into place to try and prevent fraudulent activities. The current issue of governments demanding access to keys to decrypt information is covered in Chapter 24.

■ Certificate-Based Threats

Although certificates bring much capability to security through practical management of trust, they also can present threats. Because much of the actual work is done behind the scenes, without direct user involvement, a false sense of security might ensue. End users might assume that if an HTTPS connection was made with a server, they are securely connected to the proper server. Spoofing, phishing, pharming, and a wide range of sophisticated attacks prey on this assumption. Today, the industry has responded with a high-assurance certificate that is signed and recognized by browsers. Using this example, we can examine how an attacker might prey on a user's trust in software getting things correct.

If a hacker wishes to have something recognized as legitimate, they may have to obtain a certificate that proves this point to the end-user machine. One avenue would be to forge a false certificate, but this is challenging because of the public key signing of certificates by CAs. To overcome this

problem, the hacker needs to install a false, self-signed root certificate on the end-user PC. This false key can then be used to validate malicious software as coming from a trusted source. This attack preys on the fact that end users do not know the contents of their root certificate store, nor do they have a means to validate changes. In an enterprise environment, this attack can be thwarted by locking down the certificate store and validating changes against a white list. This option really is not very practical for end users outside of an enterprise.

Stolen Certificates

Certificates act as a form of trusted ID and are typically handled without end-user intervention. To ensure the veracity of a certificate, a series of cryptographic controls is employed, including digital signatures to provide proof of authenticity. This statement aside, stolen certificates have been used in multiple cases of computer intrusions/system attacks. Specially crafted malware has been designed to steal both private keys and digital certificates from machines. One of the most infamous malware programs, the Zeus bot, has functionality to perform this task.

Stolen certificates have been implemented in a wide range of attacks. Malware designed to imitate antivirus software has been found dating back to 2009. The Stuxnet attack on the Iranian nuclear production facility used stolen certificates from third parties that were not involved in any way other than the unwitting contribution of a passkey in the form of a certificate. In less than a month after the Sony Pictures Entertainment attack became public in 2014, malware using Sony certificates appeared. Whether the certificates came from the break-in or one of the previous Sony hacks is unknown, but the result is the same.



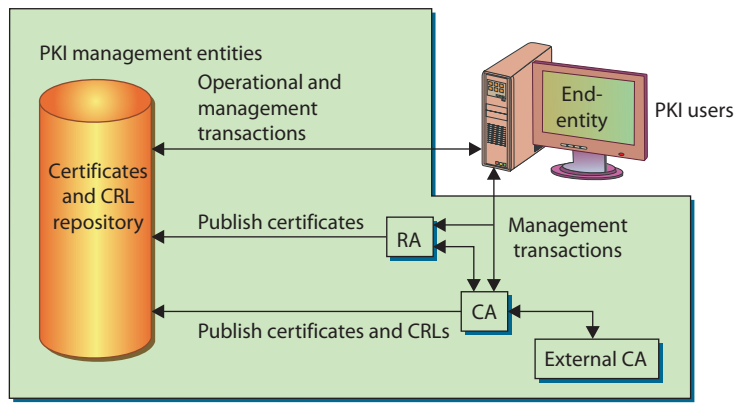
A stolen certificate and/or private key can be used to bypass many security measures. Concern over stolen SSL/TLS credentials led to the creation of high-assurance certificates, which are discussed in Chapter 17.

■ PKIX and PKCS

Two main standards have evolved over time to implement PKIs on a practical level on the Internet. Both are based on the X.509 certificate standard (discussed earlier in the “X.509” material) and establish complementary standards for implementing PKIs. PKIX and PKCS intertwine to define the most commonly used set of standards.

PKIX was produced by the Internet Engineering Task Force (IETF) and defines standards for interactions and operations for four component types: the user (end-entity), certificate authority (CA), registration authority (RA), and the repository for certificates and certificate revocation lists (CRLs). PKCS defines many of the lower-level standards for message syntax, cryptographic algorithms, and the like. The PKCS set of standards is a product of RSA Security.

The PKIX working group was formed in 1995 to develop the standards necessary to support PKIs. At the time, the X.509 Public Key Certificate (PKC) format was proposed as the basis for a PKI. X.509 includes information regarding data formats and procedures used for CA-signed PKCs, but it doesn't specify values or formats for many of the fields within the PKC. PKIX provides standards for extending and using X.509 v3 certificates and



• **Figure 7.19** The PKIX model

Tech Tip

PKI Essentials

A PKI brings together policies, procedures, hardware, software, and end users to create, manage, store, distribute, and revoke digital certificates.

for managing them, enabling interoperability between PKIs following the standards.

PKIX uses the model shown in Figure 7.19 for representing the components and users of a PKI. The user, called an *end-entity*, is not part of the PKI, but end-entities are either users of the PKI certificates, the subject of a certificate (an entity identified by it), or both. The *certificate authority* (CA) is responsible for issuing, storing, and revoking certificates—both PKCs and Attribute Certificates (ACs). The RA is responsible for management activities designated by the CA. The RA can, in fact, be a component of the CA rather than a separate component. The final component of the PKIX model is the repository, a

system or group of distributed systems that provides certificates and CRLs to the end-entities. The *certificate revocation list* (CRL) is a digitally signed object that lists all the current but revoked certificates issued by a CA.

PKIX Standards

Now that we have looked at how PKIX is organized, let's take a look at what PKIX does. Using X.509 v3, the PKIX working group addresses five major areas:

- PKIX outlines certificate extensions and content not covered by X.509 v3 and the format of version 2 CRLs, thus providing compatibility standards for sharing certificates and CRLs between CAs and end-entities in different PKIs. The PKIX profile of the X.509 v3 PKC describes the contents, required extensions, optional extensions, and extensions that need not be implemented. The PKIX profile suggests a range of values for many extensions. In addition, PKIX provides a profile for version 2 CRLs, allowing different PKIs to share revocation information.
- PKIX provides certificate management message formats and protocols, defining the data structures, management messages, and management functions for PKIs. The working group also addresses the assumptions and restrictions of their protocols. This standard identifies the protocols necessary to support online interactions between entities in the PKIX model. The management protocols support functions for entity registration, initialization of the certificate (possibly key-pair generation), issuance of the certificate, key-pair update, certificate revocation, cross-certification (between CAs), and key-pair recovery if available.
- PKIX outlines certificate policies and certification practice statements (CPSs), establishing the relationship between policies and CPSs. A policy is a set of rules that helps determine the applicability of a certificate to an end-entity. For example, a certificate for handling routine information would probably have a policy on creation, storage, and management of key pairs quite different from a policy for certificates used in financial transactions, due to the sensitivity of

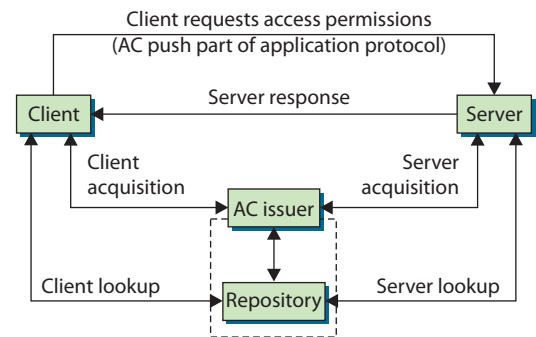
the financial information. A CPS explains the practices used by a CA to issue certificates. In other words, the CPS is the method used to get the certificate, whereas the policy defines some characteristics of the certificate and how it will be handled and used.

- PKIX specifies operational protocols, defining the protocols for certificate handling. In particular, protocol definitions are specified for using File Transfer Protocol (FTP) and Hypertext Transfer Protocol (HTTP) to retrieve certificates from repositories. These are the most common protocols for applications to use when retrieving certificates.
- PKIX includes timestamping and data certification and validation services, which are areas of interest to the PKIX working group, and which will probably grow in use over time. A timestamp authority (TSA) certifies that a particular entity existed at a particular time. A Data Validation and Certification Server (DVCS) certifies the validity of signed documents, PKCs, and the possession or existence of data. These capabilities support nonrepudiation requirements and are considered building blocks for a nonrepudiation service.

PKCs are the most commonly used certificates, but the PKIX working group has been working on two other types of certificates: Attribute Certificates and Qualified Certificates. An Attribute Certificate (AC) is used to grant permissions using rule-based, role-based, and rank-based access controls. ACs are used to implement a privilege management infrastructure (PMI). In a PMI, an entity (user, program, system, and so on) is typically identified as a client to a server using a PKC. There are then two possibilities: either the identified client pushes an AC to the server, or the server can query a trusted repository to retrieve the attributes of the client. This situation is modeled in Figure 7.20.

The client push of the AC has the effect of improving performance, but no independent verification of the client's permissions is initiated by the server. The alternative is to have the server pull the information from an AC issuer or a repository. This method is preferable from a security standpoint, because the server or server's domain determines the client's access rights. The pull method has the added benefit of requiring no changes to the client software.

The Qualified Certificate (QC) is based on the term used within the European Commission to identify certificates with specific legislative uses. This concept is generalized in the PKIX QC profile to indicate a certificate used to identify a specific individual (a single human rather than the *entity* of the PKC) with a high level of assurance in a nonrepudiation service. There are dozens of IETF Requests for Comment (RFCs) that have been produced by the PKIX working group for each of these five areas.



• Figure 7.20 The PKIX PMI model

PKCS

RSA Laboratories created the **Public Key Cryptography Standards (PKCS)** to fill some of the gaps in the standards that existed in PKI implementation. As they have with the PKIX standards, PKI developers have adopted many of these standards as a basis for achieving interoperability between different

CAs. PKCS is currently composed of a set of 13 active standards, with two other standards that are no longer active. The standards are referred to as PKCS #1 through PKCS #15, as listed in Table 7.2. The standards combine to establish a common base for services required in a PKI.

Table 7.2 PKCS Standards	
Standard	Title and Description
PKCS #1	RSA Cryptography Standard; definition of the RSA encryption standard.
PKCS #2	No longer active; it covered RSA encryption of message digests and was incorporated into PKCS #1.
PKCS #3	Diffie-Hellman Key Agreement Standard; definition of the Diffie-Hellman key-agreement protocol.
PKCS #4	No longer active; it covered RSA key syntax and was incorporated into PKCS #1.
PKCS #5	Password-Based Cryptography Standard; definition of a password-based encryption (PBE) method for generating a secret key.
PKCS #6	Extended-Certificate Syntax Standard; definition of an extended certificate syntax that is made obsolete by X.509 v3.
PKCS #7	Cryptographic Message Syntax Standard; definition of the cryptographic message standard for encoded messages, regardless of encryption algorithm. Commonly replaced with PKIX Cryptographic Message Syntax.
PKCS #8	Private-Key Information Syntax Standard; definition of a private key information format, used to store private key information.
PKCS #9	Selected Attribute Types; definition of attribute types used in other PKCS standards.
PKCS #10	Certification Request Syntax Standard; definition of a syntax for certification requests.
PKCS #11	Cryptographic Token Interface Standard; definition of a technology-independent programming interface for cryptographic devices (such as smart cards).
PKCS #12	Personal Information Exchange Syntax Standard; definition of a format for storage and transport of a user's private keys, certificates, and other personal information.
PKCS #13	Elliptic Curve Cryptography Standard. Abandoned.
PKCS #14	Pseudo-random number generation. Abandoned.
PKCS #15	Cryptographic Token Information Format Standard; definition of a format for storing cryptographic information in cryptographic tokens.

Though adopted early in the development of PKIs, some of these standards are being phased out. For example, PKCS #6 is being replaced by X.509 v3, and PKCS #7 and PKCS #10 are being used less, as their PKIX counterparts are being adopted.

Why You Need to Know the PKIX and PKCS Standards

If your company is planning to use one of the existing certificate servers to support e-commerce, you might not need to know the specifics of these standards (except perhaps for the CompTIA Security+ exam). However, if you plan to implement a private PKI to support secure services within your organization, you need to understand what standards are out there and how the decision to use a particular PKI implementation (either home-grown or commercial) may lead to incompatibilities with other certificate-issuing entities. You must consider your business-to-business requirements when you're deciding how to implement a PKI within your organization.



All of the standards and protocols discussed in this chapter are the “vocabulary” of the computer security industry. You should be well versed in all these titles, their acronyms, and their purposes and operations.

■ ISAKMP

The **Internet Security Association and Key Management Protocol (ISAKMP)** provides a method for implementing a key exchange protocol and for negotiating a security policy. It defines procedures and packet formats to negotiate, establish, modify, and delete security associates. Because it is a framework, it doesn't define implementation-specific protocols, such as the key exchange protocol or hash functions. Examples of ISAKMP are the Internet Key Exchange (IKE) protocol and IP Security (IPSec), which are used widely throughout the industry.

An important definition for understanding ISAKMP is that of the term *security association*. A security association (SA) is a relationship in which two or more entities define how they will communicate securely. ISAKMP is intended to support SAs at all layers of the network stack. For this reason, ISAKMP can be implemented on the transport layer using TCP or User Datagram Protocol (UDP), or it can be implemented on IP directly.

Negotiation of an SA between servers occurs in two stages. First, the entities agree on how to secure negotiation messages (the ISAKMP SA). Once the entities have secured their negotiation traffic, they then determine the SAs for the protocols used for the remainder of their communications. Figure 7.21 shows the structure of the ISAKMP header. This header is used during both parts of the ISAKMP negotiation.

The Initiator Cookie is set by the entity requesting the SA, and the responder sets the Responder Cookie. The Payload byte indicates the type of the first payload to be encapsulated. Payload types include security associations, proposals, key transforms, key exchanges, vendor identities, and other things. The Major and Minor Revision fields refer to the major version number and minor version number for the ISAKMP, respectively. The Exchange Type helps determine the order of messages and payloads. The Flags bits indicate options for the ISAKMP exchange, including whether the payload is encrypted, whether the initiator and responder have "committed" to the SA, and whether the packet is to be authenticated only (and is not encrypted). The final fields of the ISAKMP header indicate the Message Identifier and a Message Length. Payloads encapsulated within ISAKMP use a generic header, and each payload has its own header format.

Once the ISAKMP SA is established, multiple protocol SAs can be established using the single ISAKMP SA. This feature is valuable due to the overhead associated with the two-stage negotiation. SAs are valid for specific

Bit Position																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Initiator Cookie (8 bytes)																															
Responder Cookie (8 bytes)																															
Payload (8 bits)				Major Rev. (4 bits)				Minor Rev. (4 bits)				Exchange Type (8 bits)								Flags (8 bits)											
Message Identifier (4 bytes)																															
Message Length (4 bytes)																															

• **Figure 7.21** ISAKMP header format

periods of time, and once the time expires, the SA must be renegotiated. Many resources are also available for specific implementations of ISAKMP within the IPsec protocol.



Tech Tip

CMP Summarized

CMP is a protocol to obtain X.509 certificates in a PKI.

■ CMP

The PKIX Certificate Management Protocol (CMP) is specified in RFC 4210. This protocol defines the messages and operations required to provide certificate management services within the PKIX model. Though part of the IETF PKIX effort, CMP provides a framework that works well with other standards, such as PKCS #7 and PKCS #10.

CMP provides for the following certificate operations:

- CA establishment, including creation of the initial CRL and export of the public key for the CA
- Certification of an end-entity, including the following:
 - Initial registration and certification of the end-entity (registration, certificate issuance, and placement of the certificate in a repository)
 - Updates to the key pair for end-entities, required periodically and when a key pair is compromised or keys cannot be recovered
 - End-entity certificate updates, required when a certificate expires
 - Periodic CA key-pair updates, similar to end-entity key-pair updates
 - Cross-certification requests, placed by other CAs
 - Certificate and CRL publication, performed under the appropriate conditions of certificate issuance and certificate revocation
 - Key-pair recovery, a service to restore key-pair information for an end-entity; for example, if a certificate password is lost or the certificate file is lost
 - Revocation requests, supporting requests by authorized entities to revoke a certificate

CMP also defines mechanisms for performing these operations, either online or offline using files, e-mail, tokens, or web operations.

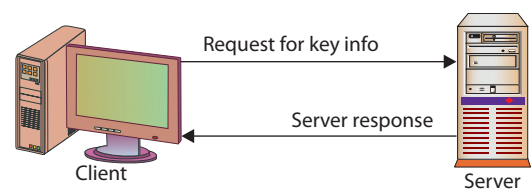
■ XKMS

The XML Key Management Specification (XKMS) defines services to manage PKI operations within the Extensible Markup Language (XML) environment. These services are provided for handling PKI keys and certificates automatically. Developed by the World Wide Web Consortium (W3C), XKMS is intended to simplify integration of PKIs and management of certificates in applications. As well as responding to problems of authentication

and verification of electronic signatures, XKMS also allows certificates to be managed, registered, or revoked.

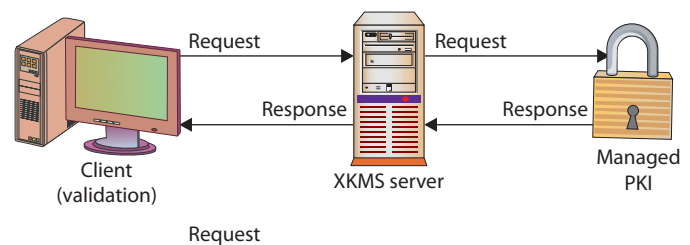
XKMS services reside on a separate server that interacts with an established PKI. The services are accessible via a simple XML protocol. Developers can rely on the XKMS services, making it less complex to interface with the PKI. The services provide for key information retrieval (owner, key value, key issuer, and the like) and key management (such as key registration and revocation).

Retrieval operations rely on the XML signature for the necessary information. Three tiers of service are based on the client requests and application requirements. Tier 0 provides a means of retrieving key information by embedding references to the key within the XML signature. The signature contains an element called a *retrieval method* that indicates ways to resolve the key. In this case, the client sends a request, using the retrieval method, to obtain the desired key information. For example, if the verification key contains a long chain of X.509 v3 certificates, a retrieval method could be included to avoid sending the certificates with the document. The client would use the retrieval method to obtain the chain of certificates. For tier 0, the server indicated in the retrieval method responds directly to the request for the key, possibly bypassing the XKMS server. The tier 0 process is shown in Figure 7.22.



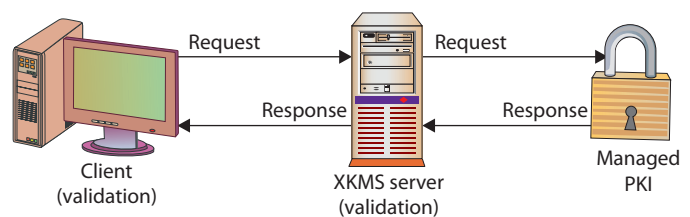
• Figure 7.22 XKMS tier 0 retrieval

With tier 1 operations, the client forwards the key-information portions of the XML signature to the XKMS server, relying on the server to perform the retrieval of the desired key information. The desired information can be local to the XKMS server, or it can reside on an external PKI system. The XKMS server provides no additional validation of the key information, such as checking whether the certificate has been revoked or is still valid. Just as in tier 0, the client performs final validation of the document. Tier 1 is called the *locate service* because it locates the appropriate key information for the client, as shown in Figure 7.23.



• Figure 7.23 XKMS tier 1 locate service

Tier 2 is called the *validate service* and is illustrated in Figure 7.24. In this case, just as in tier 1, the client relies on the XKMS service to retrieve the relevant key information from the external PKI. The XKMS server also performs data validation on a portion of the key information provided by the client for this purpose. This validation verifies the binding of the key information with the data indicated by the key information contained in the XML signature.



• Figure 7.24 XKMS tier 2 validate service

The primary difference between tier 1 and tier 2 is the level of involvement of the XKMS server. In tier 1, it can serve only as a relay or gateway between the client and the PKI. In tier 2, the XKMS server is actively involved in verifying the relation between the PKI information and the document containing the XML signature.

XKMS relies on the client or underlying communications mechanism to provide for the security of the communications with the XKMS server. The specification suggests using one of three methods for ensuring server

authentication, response integrity, and relevance of the response to the request: digitally signed correspondence, a transport layer security protocol (such as SSL, TLS, or Wireless TLS), or a packet layer security protocol (such as IPSec). Obviously, digitally signed correspondence introduces its own issues regarding validation of the signature, which is the purpose of XKMS.

It is possible to define other tiers of service. Tiers 3 and 4, an *assertion service* and an *assertion status service*, respectively, are mentioned in the defining XKMS specification, but they are not defined. The specification states they “could” be defined in other documents.

XKMS also provides services for key registration, key revocation, and key recovery. Authentication for these actions is based on a password or passphrase, which is provided when the keys are registered and when they must be recovered.

■ CEP

Certificate Enrollment Protocol (CEP) was originally developed by VeriSign for Cisco Systems. It was designed to support certificate issuance, distribution, and revocation using existing technologies. Its use has grown in client and CA applications. The operations supported include CA and RA public key distribution, certificate enrollment, certificate revocation, certificate query, and CRL query.

One of the key goals of CEP was to use existing technology whenever possible. It uses both PKCS #7 (Cryptographic Message Syntax Standard) and PKCS #10 (Certification Request Syntax Standard) to define a common message syntax. It supports access to certificates and CRLs using either the Lightweight Directory Access Protocol (LDAP) or the CEP-defined certificate query.

Chapter 7 Review

■ Chapter Summary

After reading this chapter and completing the exercises, you should understand the following about public key infrastructures.

Implement the basics of public key infrastructures

- PKI solutions include certificate authorities (CAs) and registration authorities (RAs).
- PKIs form the central management functionality used to enable encryption technologies.
- The steps a user performs to obtain a certificate for use are listed in the text and are important to memorize.

Describe the roles of certificate authorities and certificate repositories

- CAs create certificates for identified entities and maintain records of their issuance and revocation.
- CRLs provide a means of letting users know when certificates have been revoked before their end-of-life date.

Explain the relationship between trust and certificate verification

- Trust is based on an understanding of the needs of the user and what the item being trusted offers.
- Certificate verification provides assurance that the data in the certificate is valid, not whether it meets the needs of the user.

Identify centralized and decentralized infrastructures

- The three different CA architectures are hierarchical, peer-to-peer, and hybrid.
- Multiple CAs can be used together to create a web of trust.

Understand the lifecycle of certificates

- Certificates are generated, registered, and historically verified by the originating CA.
- The two main mechanisms to manage the revocation of a certificate are CRL and OCSP.

- Keys, and hence certificates, have a lifecycle; they are created, used for a defined period of time, and then destroyed.
- Certificates are handled via a certificate server and client software.
- The three classes of certificates have the following typical uses:
 - **Class 1** Personal e-mail use
 - **Class 2** Software signing
 - **Class 3** Setting up a CA

Describe public and in-house certificate authorities

- Public CAs exist as a service that allows entities to obtain certificates from a trusted third party.
- In-house certificates provide certificates that allow a firm the means to use certificates within company borders.

Identify the standards involved in establishing an interoperable Internet PKI

- PKIX and PKCS define the most commonly used PKI standards.
- PKIX, PKCS, X.509, ISAKMP, XKMS, and CMP combine to implement PKI.
- SSL/TLS, S/MIME, HTTPS, and IPSec are protocols that use PKI.

Explain interoperability issues with PKI standards

- Standards and protocols are important because they define the basis for how communication will take place.
- The use of standards and protocols provides a common, interoperable environment for securely exchanging information.
- Without these standards and protocols, two entities may independently develop their own method to implement the various components for a PKI, and the two will not be compatible.
- On the Internet, not being compatible and not being able to communicate are not options.

Describe how the common Internet protocols implement the PKI standards

- Three main standards have evolved over time to implement PKIs on the Internet.
- Two of the main standards are based on a third standard, the X.509 standard, and establish complementary standards for implementing PKIs. These two standards are Public Key Infrastructure X.509 (PKIX) and Public Key Cryptography Standards (PKCS).
- PKIX defines standards for interactions and operations for four component types: the user (end-entity), certificate authority (CA), registration authority (RA), and the repository for certificates and certificate revocation lists (CRLs).
- PKCS defines many of the lower-level standards for message syntax, cryptographic algorithms, and the like.
- There are other protocols and standards that help define the management and operation of the PKI and related services, such as ISAKMP, XKMS, and CMP.

■ Key Terms

- .cer (206)
- authority revocation list (ARL) (211)
- CA certificate (202)
- certificate (184)
- certificate authority (CA) (186)
- certificate chaining (193)
- certificate path (196)
- certificate repository (212)
- certificate revocation list (CRL) (209)
- certificate server (187)
- certificate signing request (CSR) (208)
- certification practices statement (CPS) (187)
- Common Name (CN) (203)
- cross-certification certificate (203)
- digital certificate (186)
- distinguished encoding rules (DER) (206)
- dual control (219)
- end-entity certificate (202)
- hardware security module (HSM) (217)
- hierarchical trust model (194)
- hybrid trust model (195)
- intermediate certificate (193)
- Internet Security Association and Key Management Protocol (ISAKMP) (225)
- key archiving (219)
- key escrow (220)
- key recovery (219)
- local registration authority (LRA) (188)
- offline CA (191)
- Online Certificate Status Protocol (OCSP) (211)
- online CA (191)
- P12 (207)
- P7B (207)
- peer-to-peer trust model (195)
- Personal Information Exchange (PFX) (206)
- pinning (191)
- policy certificate (203)
- Privacy Enhanced Mail (PEM) (206)
- Public Key Cryptography Standards (PKCS) (223)
- public key infrastructure (PKI) (185)
- registration authority (RA) (187)
- stapling (191)
- Subject Alternative Name (SAN) (204)
- trust model (193)
- wildcard certificates (203)
- X.509 (197)

■ Key Terms Quiz

Use terms from the Key Terms list to complete the sentences that follow. Don't use the same term more than once. Not all terms will be used.

1. The _____ is the trusted authority for certifying individuals' identities and creating an electronic document indicating that individuals are who they say they are.
2. A(n) _____ is the actual request to a CA containing a public key and the requisite information needed to generate a certificate.
3. The _____ is a method of determining whether a certificate has been revoked that does not require local machine storage of CRLs.
4. The _____ is the actual service that issues certificates based on the data provided during the initial registration process.
5. A physical device that safeguards cryptographic keys is called a(n) _____.
6. A(n) _____ is a holding place for individuals' certificates and public keys that are participating in a particular PKI environment.
7. A(n) _____ is used when independent CAs establish peer-to-peer trust relationships.
8. A(n) _____ is a structure that provides all the necessary components for different types of users and entities to be able to communicate securely and in a predictable manner.
9. _____ is the process of giving keys to a third party so that they can decrypt and read sensitive information if the need arises.
10. In a(n) _____, one CA is not subordinate to another CA, and there is no established trust anchor between the CAs involved.

■ Multiple-Choice Quiz

1. When a user wants to participate in a PKI, what component do they need to obtain, and how does that happen?
 - A. The user submits a certificate request to the CA.
 - B. The user submits a key-pair request to the CRL.
 - C. The user submits a certificate request to the RA.
 - D. The user submits proof of identification to the CA.
2. How does a user validate a digital certificate that is received from another user?
 - A. The user first sees whether their system has been configured to trust the CA that digitally signed the other user's certificate and then validates that CA's digital signature.
 - B. The user calculates a message digest and compares it to the one attached to the message.
 - C. The user first sees whether their system has been configured to trust the CA that digitally signed the certificate and then validates the public key that is embedded within the certificate.
 - D. The user validates the sender's digital signature on the message.
3. What is the purpose of a digital certificate?
 - A. It binds a CA to a user's identity.
 - B. It binds a CA's identity to the correct RA.
 - C. It binds an individual identity to an RA.
 - D. It binds an individual identity to a public key.

4. What steps does a user's software take to validate a CA's digital signature on a digital certificate?
 - A. The user's software creates a message digest for the digital certificate and decrypts the encrypted message digest included within the digital certificate. If the decryption performs properly and the message digest values are the same, the certificate is validated.
 - B. The user's software creates a message digest for the digital signature and encrypts the message digest included within the digital certificate. If the encryption performs properly and the message digest values are the same, the certificate is validated.
 - C. The user's software creates a message digest for the digital certificate and decrypts the encrypted message digest included within the digital certificate. If the user can encrypt the message digest properly with the CA's private key and the message digest values are the same, the certificate is validated.
 - D. The user's software creates a message digest for the digital signature and encrypts the message digest with its private key. If the decryption performs properly and the message digest values are the same, the certificate is validated.
5. Why would a company implement a key archiving and recovery system within the organization?
 - A. To make sure all data encryption keys are available for the company if and when it needs them
 - B. To make sure all digital signature keys are available for the company if and when it needs them
 - C. To create session keys for users to be able to access when they need to encrypt bulk data
 - D. To back up the RA's private key for retrieval purposes
6. Within a PKI environment, where does the majority of the trust actually lie?
 - A. All users and devices within an environment trust the RA, which allows them to indirectly trust each other.
 - B. All users and devices within an environment trust the CA, which allows them to indirectly trust each other.
 - C. All users and devices within an environment trust the CRL, which allows them to indirectly trust each other.
 - D. All users and devices within an environment trust the CPS, which allows them to indirectly trust each other.
7. Which of the following properly describes what a public key infrastructure (PKI) actually is?
 - A. A protocol written to work with a large subset of algorithms, applications, and protocols
 - B. An algorithm that creates public/private key pairs
 - C. A framework that outlines specific technologies and algorithms that must be used
 - D. A framework that does not specify any technologies but provides a foundation for confidentiality, integrity, and availability services
8. Once an individual validates another individual's certificate, what is the use of the public key that is extracted from this digital certificate?
 - A. The public key is now available to use to create digital signatures.
 - B. The user can now encrypt session keys and messages with this public key and can validate the sender's digital signatures.
 - C. The public key is now available to encrypt future digital certificates that need to be validated.
 - D. The user can now encrypt private keys that need to be transmitted securely.

9. Why would a digital certificate be added to a certificate revocation list (CRL)?
 - A. If the public key had become compromised in a public repository
 - B. If the private key had become compromised
 - C. If a new employee joined the company and received a new certificate
 - D. If the certificate expired
10. How can users have faith that the CRL was not modified to present incorrect information?
 - A. The CRL is digitally signed by the CA.
 - B. The CRL is encrypted by the CA.
 - C. The CRL is open for anyone to post certificate information to.
 - D. The CRL is accessible only to the CA.

■ Essay Quiz

1. You are the Information Security Officer at a medium-sized company (1500 employees). The CIO has asked you to explain why you recommend using commercial PKIs rather than implementing such a capability in-house with the software developers you already have. Write three succinct sentences that would get your point across and address three key issues.
2. Describe the pros and cons of establishing a key archiving system program for a small- to medium-sized business.
3. Why would a small- to medium-sized firm implement a PKI solution? What business benefits would ensue from such a course of action?
4. Describe the steps involved in verifying a certificate's validity.
5. Describe the steps in obtaining a certificate.
6. Compare and contrast the hierarchical trust model, peer-to-peer trust model, and hybrid trust model.

Lab Projects

• Lab Project 7.1

Investigate the process of obtaining a personal certificate or digital ID for e-mail usage. What

information is needed, what are the costs, and what protection is afforded based on the vendor?

• Lab Project 7.2

Determine what certificates are registered with the browser instance on your computer.